Assignment 1: Simulated Annealing

Michael Toce

January 19, 2016

1 Travelling Salesman

The travelling salesman is a problem in finding the shortest total length of a salesman who must travel between N cities. The problem we run into is that as N increases, the time to compute the shortest possible total length increases exponentially:

$$t \propto e^{constant*N} \tag{1}$$

Instead, we can approach this problem by thinking about a gas in a box, whose molecules have an energy proportional to the temperature in that box. Assuming the particles do not collide, each particle makes up a section of the given energy within the box. This is known as the Boltzmann distribution:

$$\frac{dP}{dE} \propto e^{\frac{-E}{kT}} \tag{2}$$

As the temperature decreases inside the box, the molecules freeze in place. In this metaphor, the molecules are the cities and the difference in energy is the difference in total path lengths between two permutations.

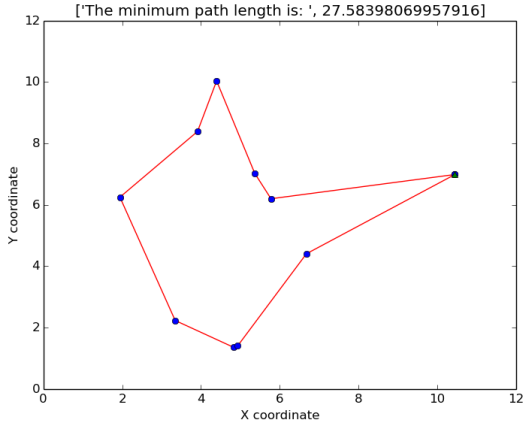

Figure 1: This figure denotes the placement of random cities along the x-y grid. The lines represent the minimum total path the salesman takes with the green triangle representing the city he starts and ends.

1.1 The Robust Method

However, the robust method can be used on a small number of cities. For my code, 10 cities were the maximum that could be used in a timely manner. With this method, the code simply looked at every single possible iteration of cities and plotted the one with the minimum total path length. This path is shown in the graph above and to the right.

The minimum total path length I found from the robust method from my list of cities was 27.584.

However, with 10 cities being the maximum number the computer could handle, seeing as:

$$N_permutations = N_cities! \quad (3)$$

, and the purpose of the code is to check each permutation to find a specific one, the number of checks would easily get to a range where no matter how efficient the code, the computing power would not be able to achieve success.

1.2 The Simulated Annealing Method

However, with the simulated annealing method, the number of cities is not a problem whatsoever, since the we can zero in on a local minimum (possibly the global minimum). By using the philosophy of equation (2) we can choose to accept a new minimum path even if it is greater by a simple probability function which scales with the temperature. By slowly decreasing the temperature, the probability will become less and less likely to accept the new path if it is greater, so only local minima will therefore be accepted. This trend of probability vs. time would slowly decrease.

For the results of the simulated annealing problem, the program could run much faster for a given number of iterations when the cities were higher. To run 10 cities and get an accurate result, it would take the system 0.52 seconds to run. Meanwhile, with the robust method, it would take 0.58 (but over a minute to graph). So 10 cities is where the two really start drifting. At 5 cities, the robust method took 0.117 seconds while the simulated annealing took 1.07 seconds. Meanwhile, at 13 cities, the robust method took an indeterminate number of seconds while the simulated annealing took 1.02 seconds.

When annealing for my salesman, it seemed as though only at the very end were points not considered by via the probability, although a close to perfect result was always the case when iterations were high. If I tried to crank T down faster in order to make up for this, somehow my results were less accurate. This was an interesting results as you would expect to yield better results.

So, it seems as if cranking T down by about half each 100 or so permutations yields the best results.

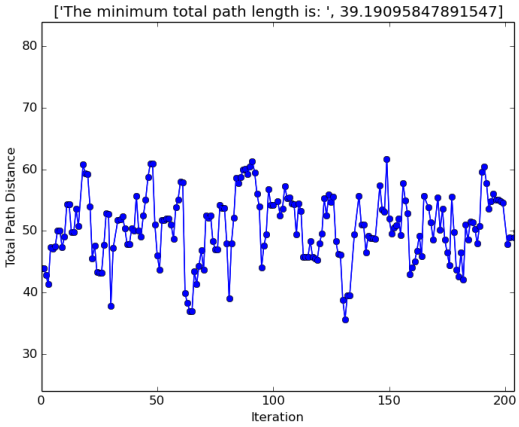

Figure 2: Cranking T down by 90 percent each 100 iterations

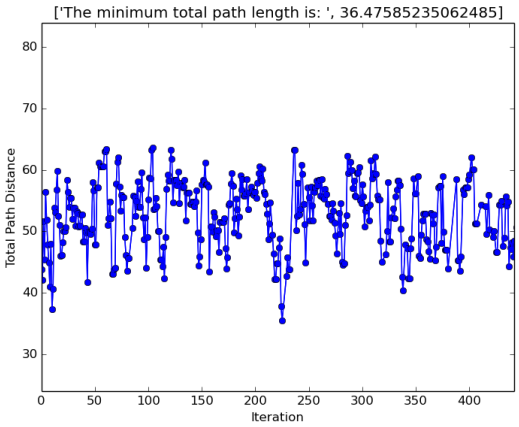

Figure 3: Cranking T down by 50 percent each 100 iterations

Those results however varied drastically from each running of the program. One time my result would be 36, another time 28. The result largely mattered on how many iterations of code was done in between each change in temperature. The larger number of iterations, the better the result. However, variation still occurred. I wonder if it is because the parameters of the annealing function were not properly adjusted to account for this, or if the code in itself was not annealing to an exponential curve like it should.

1.3 With Tolls and other Roadblocks

With tolls included, there would also be a monetary factor to include. It is important whether to determine the scale of importance of these two factors, and the scale of the probability should also scale with the difference in money spent as well.

For speed limits, instead of calculating the total distance travelled, you would simply find the time it takes to travel between cities instead of the distance, based on the speed limit of each road (you could even assume speeding by a certain percentage).

Although it took me so long to debug my code, there were no additions with these results.