Previously accomplished:

Graphics set up, greedy solution generator, cross breed, best solution distance: 19608.

Monday, February 15, 2016:

1. Increase Population size from 43 to approximately 976. 43 solutions are greedy solutions and the rest is randomly generated.
2. File i/o random indexes into R.py so that the population remains the same each time we run main.

Since the breeding population is sorted from the shortest distance to the longest distance, the more optimal solution should have a better chance at being selected for breeding. In this study, linear and exponential function will be explored as models for the probability.

Let linear function denote the probability of solution with index x in the population of size getting picked for cross breeding, since the sum of the probabilities for the each individual solution in the entire breeding population needs to equal 1, constants and can be found given the following integral,

The probability of the last solution of index n being picked should approach 0, given that an expression of in terms of and can be established as

Given , and and be found with the system of equations shown.

Let exponential function denote the probability function. Then,

Similar to the linear model, an expression of in terms of and and be found as

Again and and be found given .

For example, a breeding population of size 781 would give and 739 with 6 significant figures for and .

As seen from the graph above, the optimal solution has a chance ten times higher in the exponential model than the linear model, which allows the greedy solutions that are locally optimized to be picked far more often than the random solutions. However, the disadvantage is the lack of randomness as the solutions towards to end have probabilities approaching 0.

It is worth noting that although the solving the integral approximates the model quite well, realistically the model needs to be fine tuned since is infinitely small. In reality, we want the Riemann sum to be 1 instead of the definite integral.

Tuesday, February 15

1. Finished probability distribution, translated into python and attempted to breed.

Starting:

best: 19608 worst: 96341 average: 78160 range: 76733 size: 976

Stats for exponential probability distribution model cross breeding, no mutation, 100 iteration:

best: 19097 worst: 27949 average: 21247 range: 8852 size: 899

best: 19390 worst: 28021 average: 21330 range: 8631 size: 896

best: 19323 worst: 27565 average: 21230 range: 8242 size: 936

Stats for exponential probability distribution model cross breeding, no mutation, 10 iteration:

best: 19571 worst: 29012 average: 22665 range: 9441 size: 930

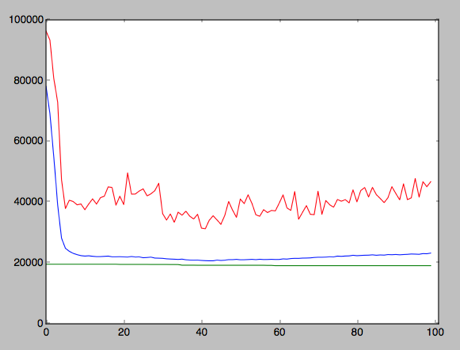
best: 19548 worst: 29000 average: 22351 range: 9452 size: 922

best: 19571 worst: 30492 average: 22408 range: 10921 size: 932

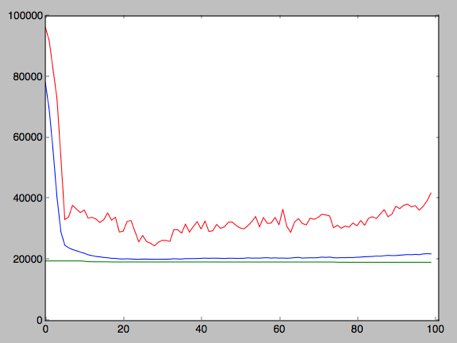
exponential probability distribution model, cross breeding, 100 iteration:

Mutations:

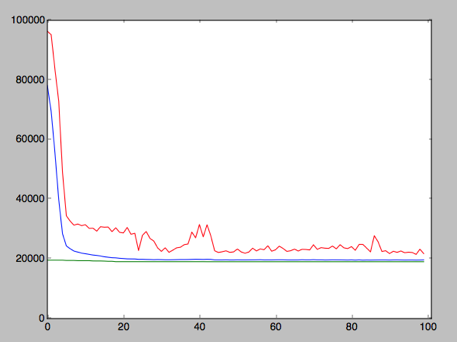
1. 1 swap mutation in solution:
   1. 70% of parent1
      1. best: 19125 worst: 46482 average: 23151 range: 27357 size: 1178



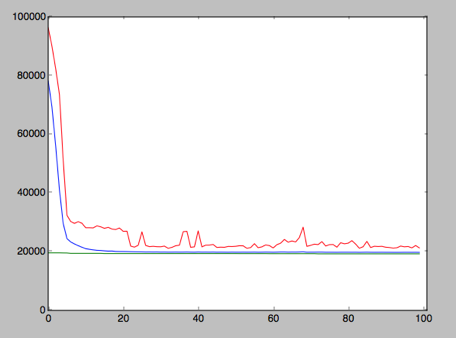
* 1. 50% of parent1
     1. best: 19160 worst: 40477 average: 21981 range: 21317 size: 1164



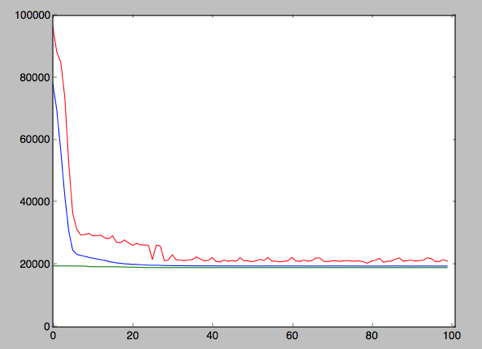
* 1. 30% of parent 1
     1. best: 19071 worst: 22588 average: 19648 range: 3517 size: 869



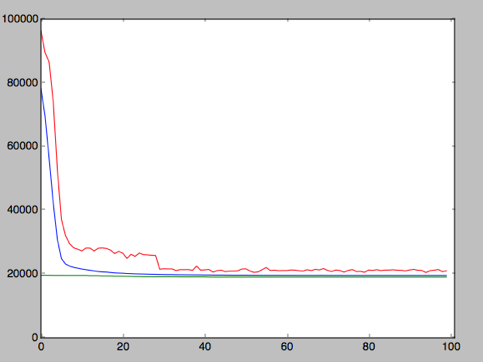
* 1. 15% of parent1
     1. best: 19281 worst: 21533 average: 19822 range: 2252 size: 825



1. 1 swap mutation of child:
   1. 30%
      1. best: 19071 worst: 21265 average: 19588 range: 2194 size: 820



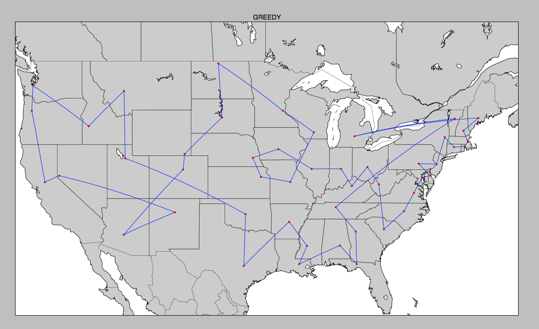
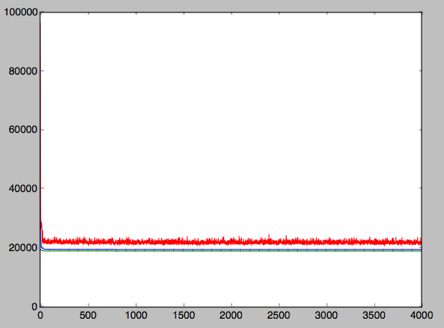
* 1. 15%
     1. best: 19071 worst: 20957 average: 19567 range: 1886 size: 790



Wednesday February 17

4000 iteration 30% 1 swap child mutation exponential probability half cut cross breed

best: 19031 worst: 22087 average: 19552 range: 3056 size: 852



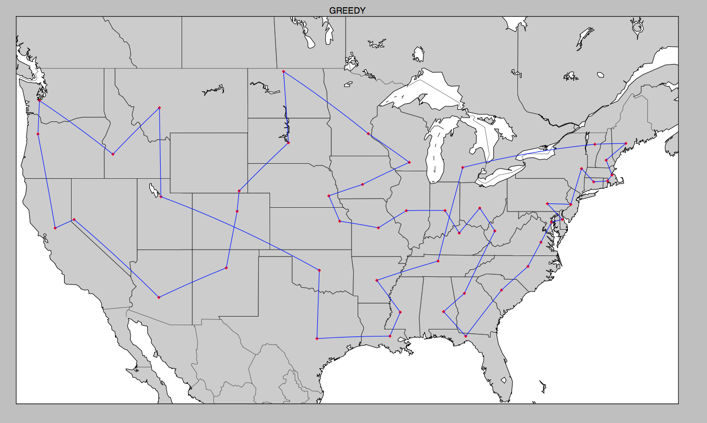
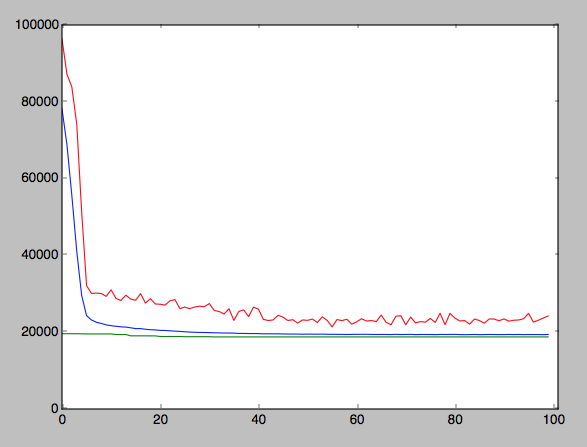
Thursday, February 18

Developed new mutation called reverse. Randomly chooses segments of code of length between 2-5 and reverse the order in attempt to untie knots.

100 iteration, exp prob, rev2-5 mut on child 30%, half cut cross breed

best: 18772 worst: 23654 average: 19391 range: 4882 size: 840

it has only been 100 iterations and the best solution has already improved by nearly 1000, and the average is already lower than the best solution in original population. Graph looks similar to the previous mutation



The graph is two knots away from passing the stupidity test!!!

Friday February 19

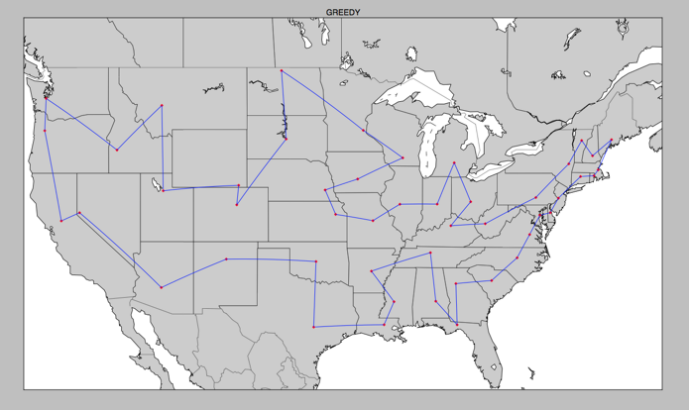
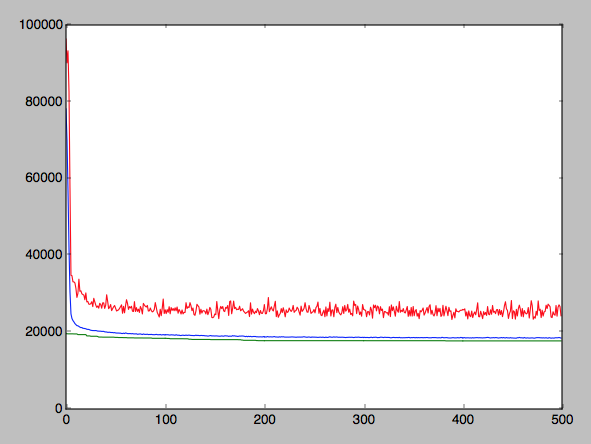
100 iteration exp prob, rvrs2-10 mut on child 30%, half cut cross breed.

best: 18501 worst: 25732 average: 19291 range: 7231 size: 875

500 iteration exp prob rvrs2-19 mut on child 30%, half cut cross breed.

best: 17737 worst: 25360 average: 18481 range: 7623 size: 886

OPTIMAL SOLUTION PASSED THE STUPIDITY TEST.



500 iteration exp prob rvrs2-25 mut on child 30%, half cut cross breed.

best: 17591 worst: 27628 average: 18633 range: 10037 size: 924

We see that as the possible segment for reversal increase, the best solution improves but the range increases and the worst solution stays high, though the average is still low.

Wednesday February 24

Worked on weave() function using recursion, lots of issues.

Thursday February 25

Day 8 met with Mr.G.

Dicsussed future goals: potential blinking nodes, simulated annealing

Simulated annealing: start from very random solution, and gradually decrease the randomness through picking chunks of solution, randomize the chunks and see if it improves.

New direction for weave(), mark the cities already in the child as false and ignore any false cities instead of removing them as cross breed happen. This result in less runtime as removing elements costs a lot of time.

New mutation, pick a random chunk and move it to a random place.

Settled on exp probability, no need to change n.

New population idea, different types of greedy algorithm with not only the shortest city but the second shortest, third shortest and so on.

Fix Bismarck coordinates.