GA Design

The best way to explain the design of my GA is to walk through a simple example. In this case, there will be 4 houses.

Chromosome Design

Here is the layout for one chromosome:

Truck 1				Truck 2			
h1	h2	h3	h4	h1	h2	h3	h4

Each gene (h1, h2… etc.) represents a house, and the value of h_n represents the order it is visited. A 0 means the truck doesn't visit that house.

Example:

0	2	1	0	1	0	0	2

Truck 1 visit order → h3, h2

Truck 2 visit order → h1, h4

Chromosome Design

The first chromosome would divide the houses by which warehouse it started closer too. The additional chromosomes were versions of this with the visitation order shuffled and mutations to cause switches in which truck would visit. This created a dynamic where the initial chromosomes were not so random that they would have a truck going back and forth across the map from corner to corner.

Mutation

Mutation selects a random gene in a chromosome, then changes its value either:

1) from 0 to the end of the list for that truck
2) from non-zero value to zero

The house represented by that gene is then has the opposite change occur in the other truck.

Any gaps in visitation order (ex: truck has [1 0 0 4]) is fixed by a function called “shift”, that shifts values with gaps down to their proper order (ex: [1 0 0 4] → [1 0 0 2]).

Although mutations (and fixed overlap in crossover) always move changed values to end of sequence, this is fixed through the process of many generations. When the smaller value is altered in the future, the larger value from the current mutation will shift down.

Example:

0	2	1	0	1	0	0	2

Mutation 1

0	2	0	0	1	0	3	2

Shift

0	1	0	0	1	0	3	2

Mutation 2

0	1	0	2	1	0	3	0

Shift

0	1	0	2	1	0	2	0

Crossover

Crossover selects a random section in half a chromosome (one truck’s section) and replaces it with the same section from another, random chromosome.

This can cause several issues, the first one addressed is overlap of house order within each truck. This is fixed by the “fix_order” method, which randomly chooses an overlapping gene and moves it to the back of the visit order. The next issue are houses with overlapping 0s or non-zero values. This is fixed by the “fix_overlap” method, which once again identifies these points and randomly chooses one of the bad genes to change so the chromosome is valid.

Example:

2	1	0	0	0	0	2	1
0	2	0	1	1	0	2	0

Random selection = h3

Crossover

2	1	0	1	0	0	2	0

Fix Order

3	1	0	2	0	0	2	0

Fix Overlap (none in this example, but happens with many houses)

3	1	0	2	0	0	2	0

Shift

3	1	0	2	0	0	1	0

This crossed chromosome is now valid.

The issue with all these additional randomizations is it can cause some crossovers to make very big, unpredictable changes. This can be good or bad, which will be covered in depth in the results section.

Switch

This operation switches the order value of two non-zero indices in a single truck.

Example:

0	2	1	0	1	0	0	2

Switch 1

0	1	2	0	1	0	0	2

Fitness

Fitness is a function of total distance traveled by both trucks in a chromosome. For sake of proper function in the roulette wheel, the fitness is:

$$fitness = \frac{10^{100}}{distance^{100}}$$

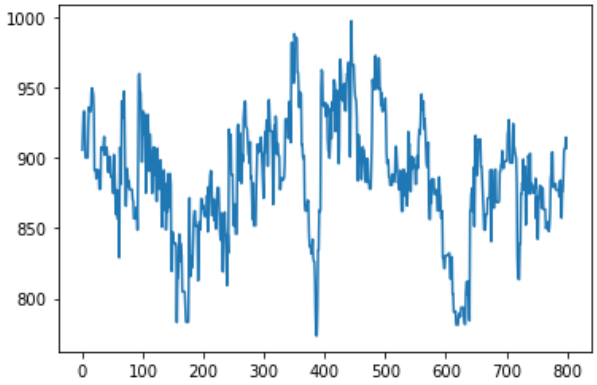
It is set up like this because:

1) The inverse of distance causes larger distances to have lower fitness, reducing their chances of being chosen compared to chromosomes with smaller distances.

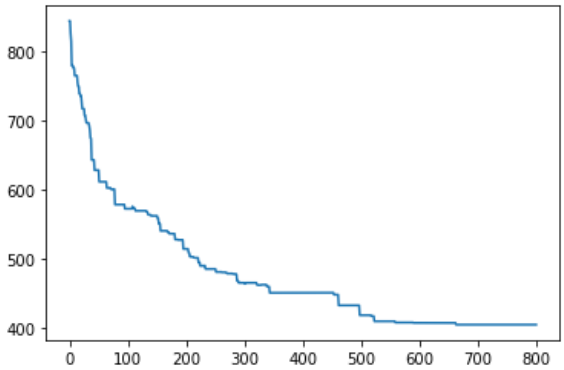
2) Initially, the difference in percentages in the roulette wheel was very small, so the GA wasn't selecting good chromosomes more often than bad ones. This caused the GA not to optimize. The distance is now multiplied many times to increase the difference in fitnesses for larger difference in fitness ratio.

Example:

Fitness = 1/d



Fitness = $10^{100}/ d^{100}$



Results

The ratio of mutation, crossovers and switches had a huge effect on the output.

I started with only mutations and crossovers.

Mutations were faster operations and had more consistent improvements with less deviation in best distance between simulations. The small changes allowed for a consistent improvement of chromosomes per generation, but these small changes could result in getting stuck at local minima.

Runs with high crossover rates had large deviations because of the various additional steps required to make many crossovers valid. This was both good and bad. Good in that the large changes in chromosomes sometimes helped prevent the GA from getting stuck at local minima. Bad in that it's hard to tweak an already good chromosome with only large changes which can result in overshooting minima. I found that these trials did best with around a 30% crossover rate.

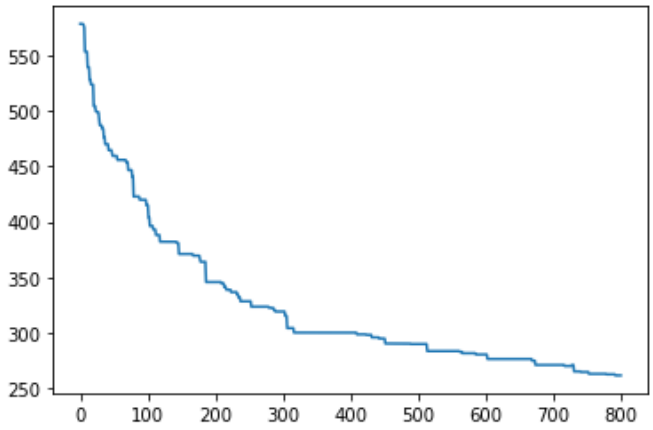
After running trails of just mutations and crossovers, I decided to try the switch operator. The inspiration for switches was that mutations only switched houses between trucks and crossovers had many changes, so it seemed like a good idea to try switching visitation orders within trucks.

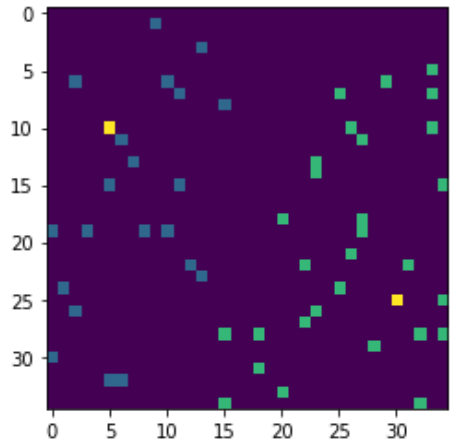
This worked great. In my tests I found that doing a mix of mutations and switches created a great dynamic of controlled randomness that rarely got stuck at local minima. The best ratio was 80% switches and 20% mutations. I tried various ratios with all three operators, but it wasn't as good as with just switching and mutations.

For reference (800 generations, 10 chromosomes selected per generation):

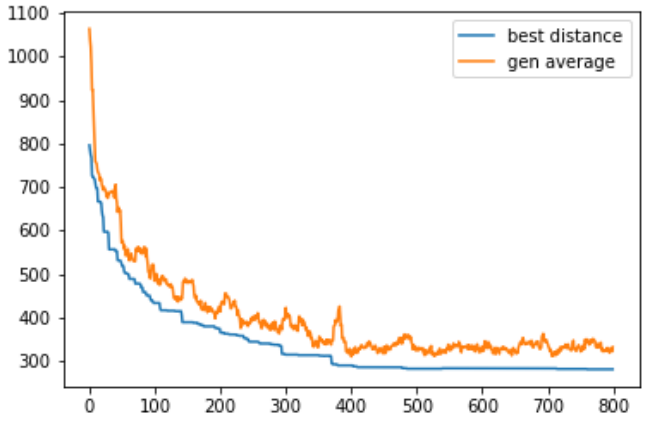
Operation Ratio	Only Mutations	70% Mutation 30% Crossover	20% Mutation 80% Switch
10 Run Best Distances	412.87 427.45 445.81 448.64 475.45 469.70 446.14 416.41 486.64 426.56	395.49 493.9 427.27 415.96 424.04 460.79 451.21 461.19 416.44 437.57	283.56 307.67 270.04 261.19 274.18 319.77 294.53 312.39 276.03 261.2
Average	445.57	438.39	286.06
Lowest	412.87	395.49	261.19

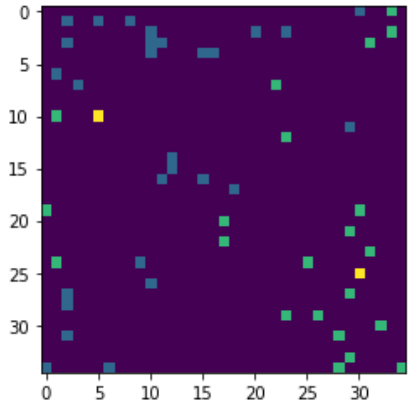
The best run I had was ~= 254 distance using 20% mutation and 80% switch.

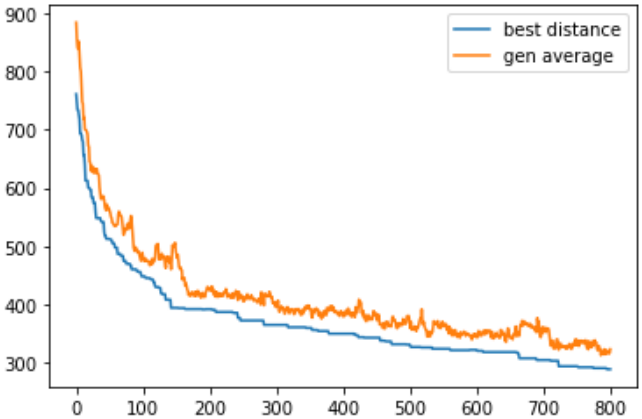


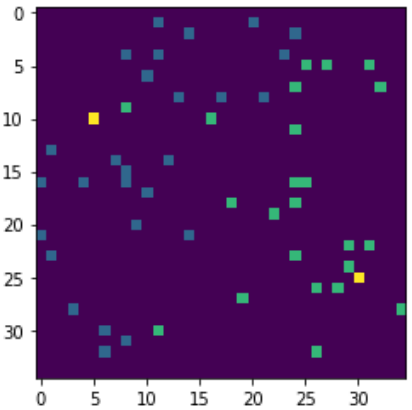


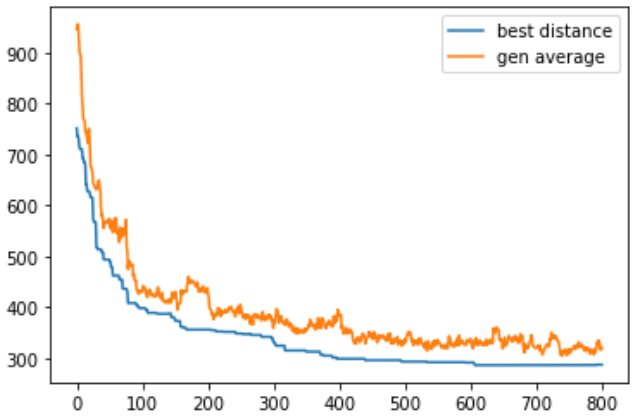
5 Test Runs with Graphs

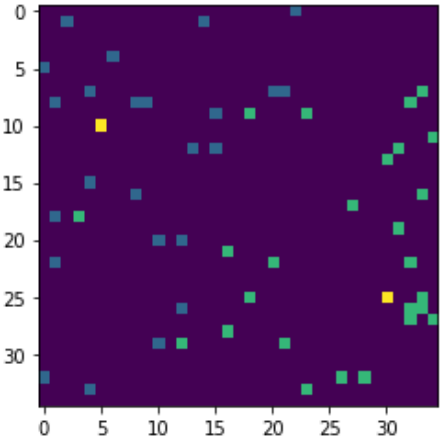


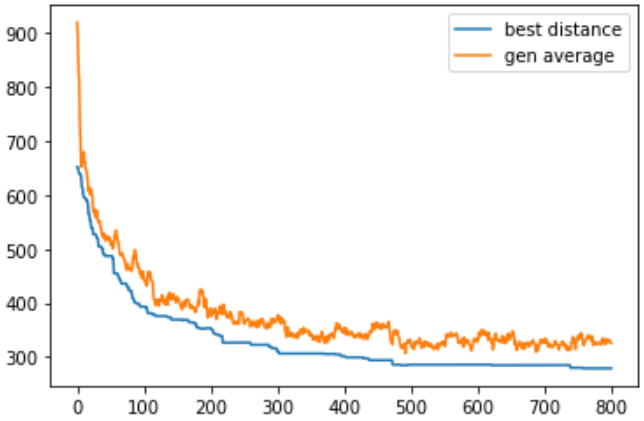


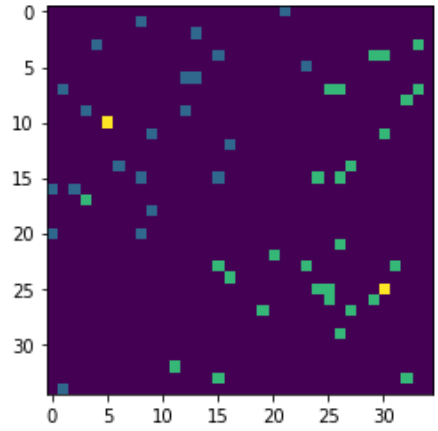


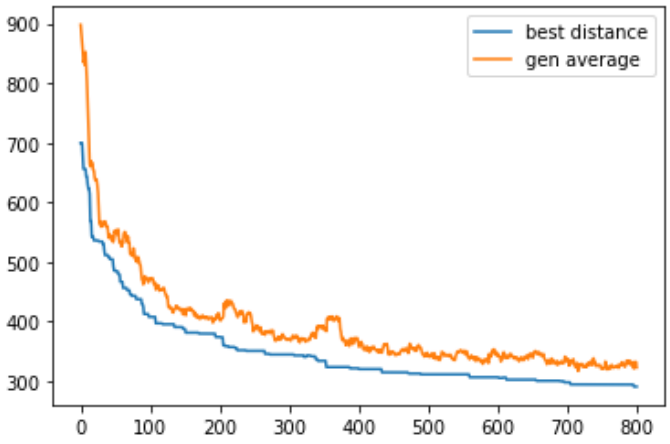


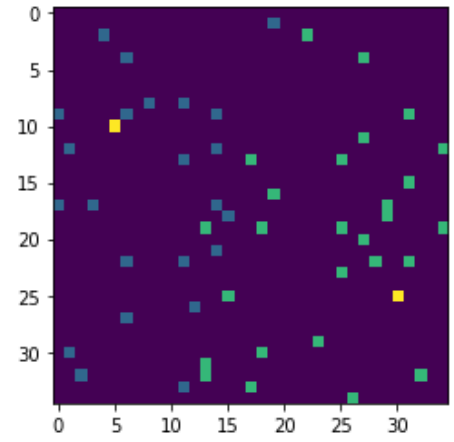

best distance
gen average
900
800
700
600
500
400
300
0
100
200
300
400
500
600
700
800


0
5
10
15
20
25
30


best distance
gen average
900
800
700
600
500
400
300
0
100
200
300
400
500
600
700
800


0
5
10
15
20
25
30


best distance
gen average
900
800
700
600
500
400
300
0
100
200
300
400
500
600
700
800


0
5
10
15
20
25
30

GA Comparison

GAs are worse than other methods (such as exhaustive searches) because they are not guaranteed to find the optimal solution. However, they are much more computationally and time efficient. Additionally, not all other methods can escape local minimums, which GAs are capable of.

Conclusion

Overall, I was very happy with the results of my GA. The random initialized chromosomes usually had 700 – 1000 distance, and within 6 – 7 seconds (the time for 800 generations with 10 chromosomes) that was consistently down to 300, many times even less. If ran for a longer time, I'm confident the GA would find the optimal solution much faster than an exhaustive search.

I was initially surprised at the ineffectiveness of crossover. However, after taking a step back from the problem and not just considering traditional GA operations, it made sense. If I had no knowledge of GAs and was asked to find the optimal combination in this situation, I would say that I should try switching the visitation order of 2 houses or switch which truck visits a house when trying new combinations.

I wouldn't think to take a random chuck of houses and orders and switch them because that could cause huge leaps in visitation order and how many houses a specific truck must visit. With the additional changes required to make a crossover valid, it's just too random with too large of changes per operation to efficiently optimize with. It can be useful when stuck in local minima, but can take a large number of crossovers to find a child that is fit enough to have a chance of making the next generation and that child may converge to another local minima.