Nicholas Peterzell
COGS 160
5/21/2021

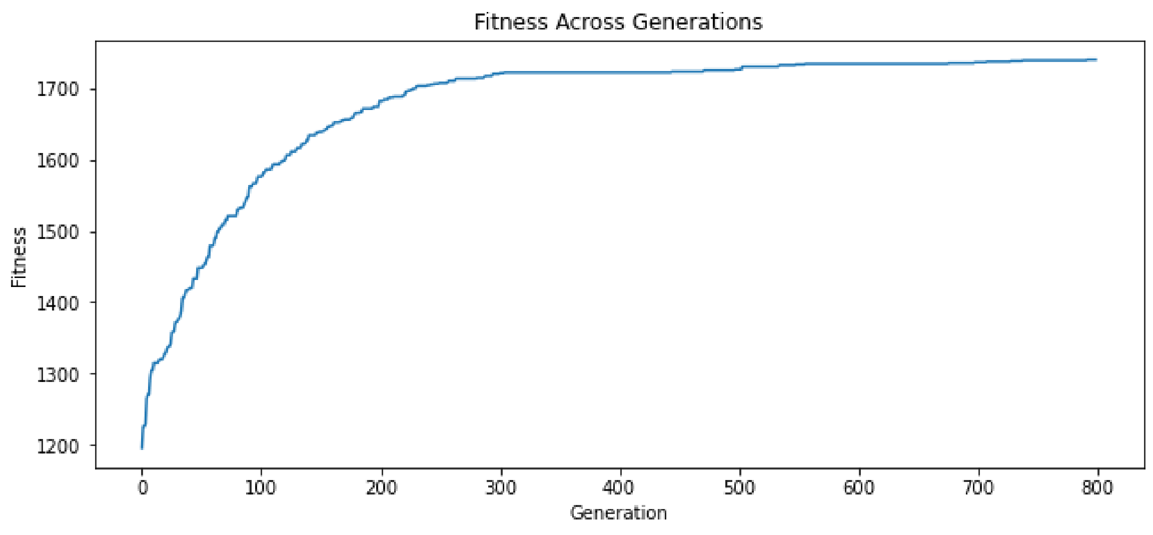
Assignment 2 Report

This report contains the results I found after solving a 01-Knapsack problem with genetic algorithms, testing 3 different crossover functions and one mutation function, and comparing the results to a Dynamic Programming solution. It also contains the results of solving a 20-city travelling salesman problem using a genetic algorithm function utilizing one crossover function and one mutation function.

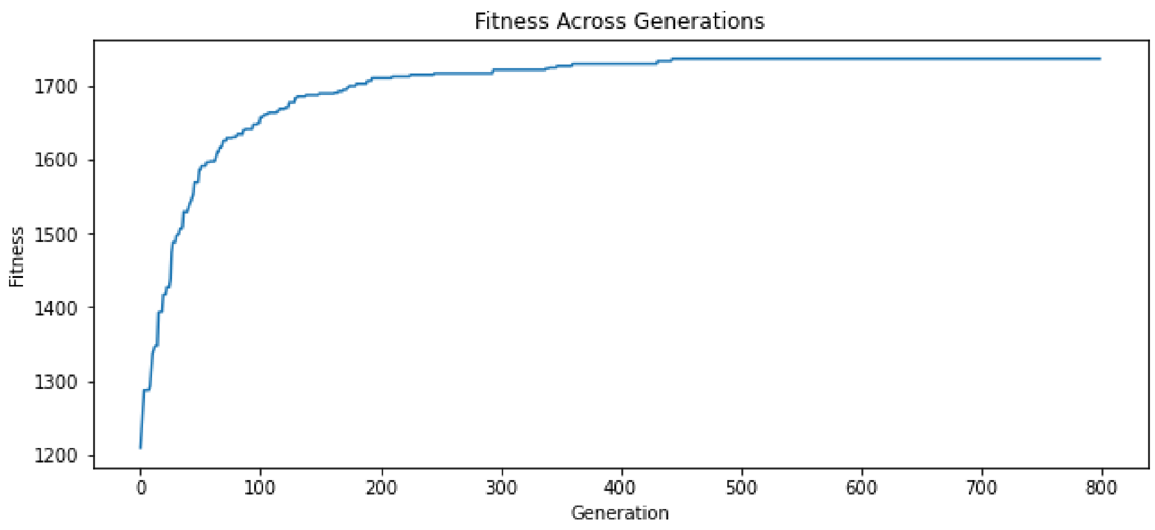
Crossover Function Used	**Average Final Fitness**
Split	1741.6
Random Third	1740.2
Ordered	1745.8

Dynamic Programming	1757.0

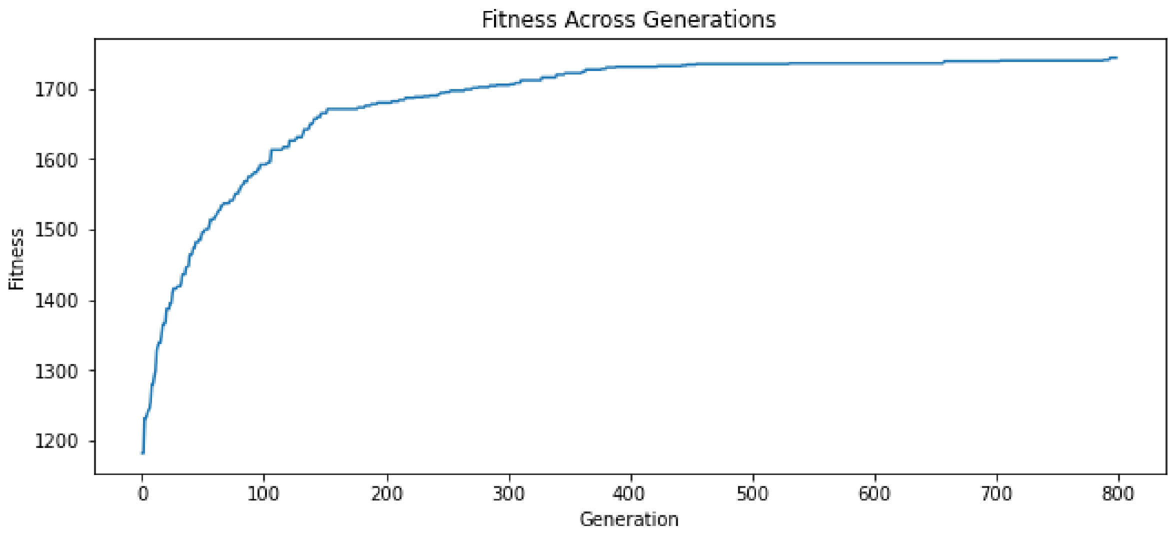
Split Crossover Plot



Random Third Crossover Plot



Ordered Crossover Plot



Description of mutation and crossover functions

For the knapsack problem, the mutation function had a mutation rate of 20%, meaning that each individual in each generation had a 20% chance of being mutated. If an individual was selected to be mutated, one random choice in that individual's list of choices would be changed from a 0 to a 1 ("putting it in" the knapsack) or a 1 to a 0 ("taking it out" of the knapsack).

The crossover functions were designed to create two children from two parents, and worked as follows:

- Split Crossover: This crossover function selects a random point in the list of choices for the problem; then each parent's list of choices is split on that point and merged with the other parent's choices on the opposite side of the point. For example, the list of choices on the left side of the point for parent 1 would be connected with the list of choices on the right side of the point for parent 2, and the reverse is also carried out, returning two children.

- Random Third Crossover: This crossover function randomly selects an unordered third of the choice positions for the problem, and then crosses over the choices in those positions between the parents; the selected choices from parent 1 are crossed over to parent 2, and the same choices are crossed over from parent 2 to parent 1, producing two children.

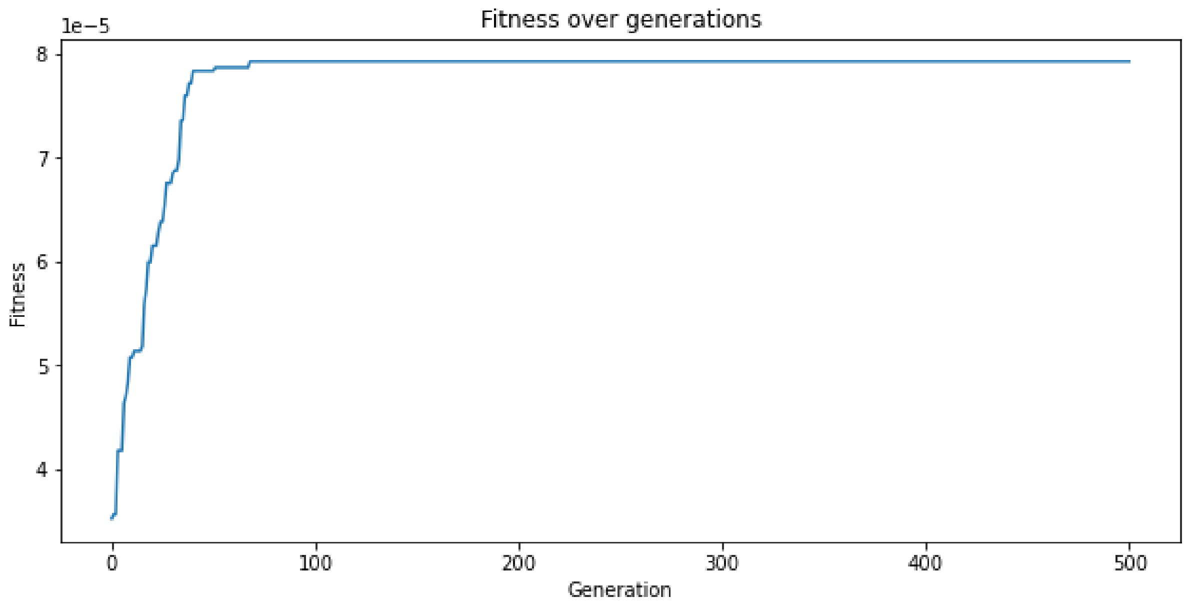
- Ordered Crossover: This crossover function selects a continuous subset of positions, of random size, from the list of choices for the problem and then crosses over the parents on this subset. For instance, the choices in that subset for parent 1 are transferred over to the same positions in parent 2, and the choices in that subset in parent 2 are transferred over to the same positions in parent 2. This process produces two children.

Discussion of mutation and crossover functions

The mutation function that I used seems to have performed well; across all crossovers, the fitness score was very high, and the algorithm did not converge prematurely. I believe that having my mutation function only make minor adjustments to individuals prevented the mutation from causing any drastic changes to the solutions when the algorithm was closing in on a high fitness score. Looking at my crossover functions, all of them seemed to have worked very well, since they all scored highly; within a very close margin of the dynamic programming method's fitness score. My guess as to why this is would be that my functions caused a lot of variation in the solutions; the split crossover function picked a random point to split on, the random third crossover function picked a completely random set of choice positions to cross over, and the ordered function selected a random continuous list of choice positions to cross over. I believe this level of randomness in the crossovers allowed for the algorithm to explore solutions that varied significantly, giving it a better chance of finding a solution with a high fitness score.

Travelling Salesman Fitness Plot

Initial distance: 27394.70
Final distance: 11771.83

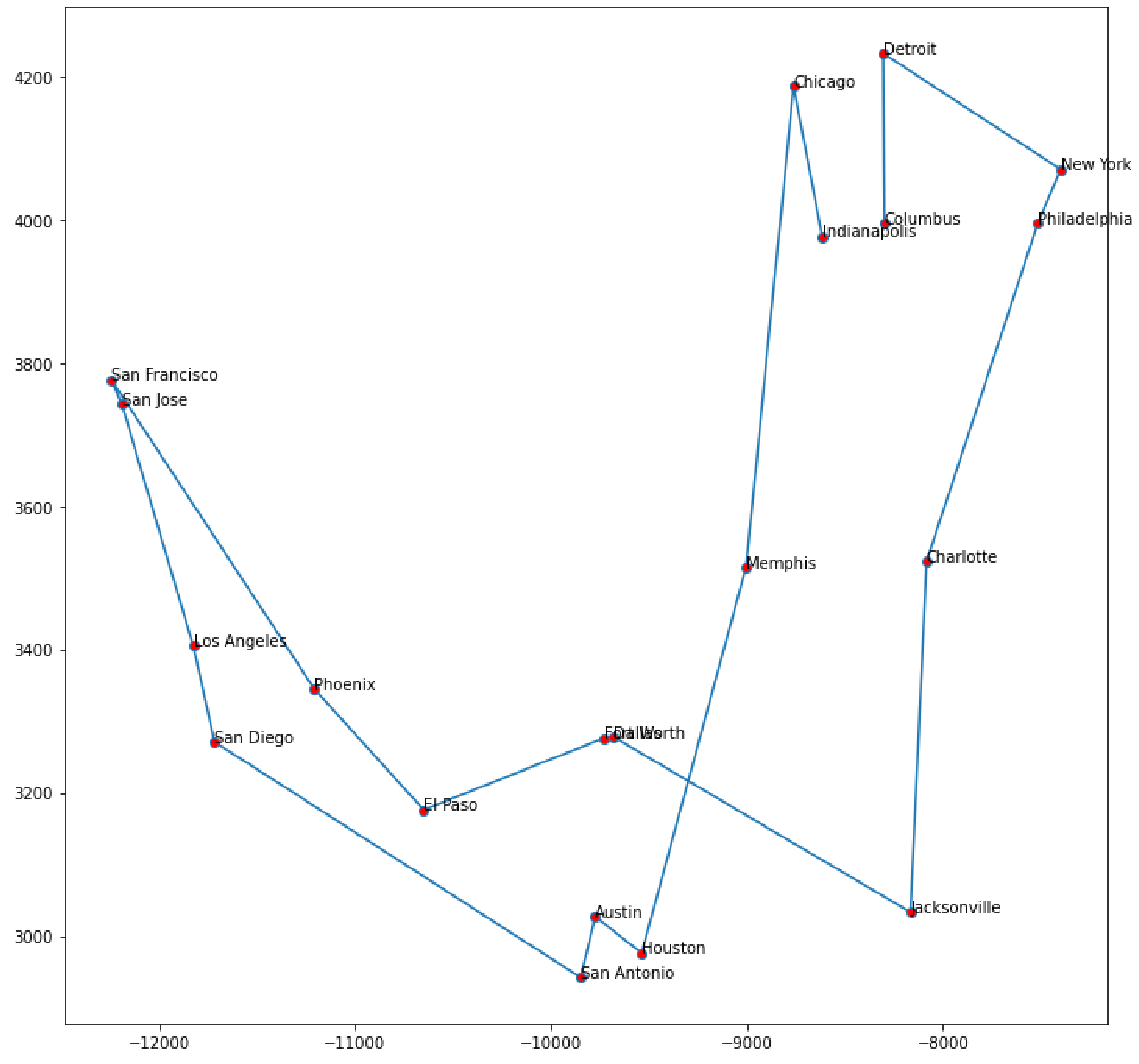


Travelling Salesman Final Route

City List:

Columbus - Detroit - New York - Philadelphia - Charlotte - Jacksonville - Dallas - Fort Worth - El Paso - Phoenix - San Francisco - San Jose - Los Angeles - San Diego - San Antonio - Austin - Houston - Memphis - Chicago - Indianapolis

Plot:


Detroit
Chicago
New York
Columbus
Philadelphia
Indianapolis
San Francisco
San Jose
Memphis
Charlotte
Los Angeles
Phoenix
San Diego
Fort Worth
Dallas
El Paso
Jacksonville
Austin
Houston
San Antonio