Assignment 2: Genetic Algorithms with the Travelling Salesman Problem

April 23, 2018

Aryana Collins Jackson

Applied Machine Learning
Dr. Ted Scully

Introduction

In this assignment, a genetic algorithm was used to solve the travelling salesman problem. Two datasets were used, one containing information for fifteen cities and one containing information for forty-nine cities. The two .txt files contained distance matrices with the distances between each city.

Methodology and Code

In order to solve the travelling salesman problem, an algorithm was set up such that individuals were created from the file, a population was created, children and mutations were created from that population, and generations were created through iterations to find the individual with the lowest overall distance. Throughout this process, five different parameters were changed to optimise the solution: parent selections, crossovers, mutations, number of generations, and population size.

Parent Selections

Within the code, there are three different functions for the parent selection from the population.

Round 1
The first round of parent selection involved pulling just two individuals from the population and choosing the one with the best fitness (lowest overall distance). This was done twice in order to choose two parents.

Round 2
The second round of parent selection involved selecting 20% of the population at random. The individual with the best fitness was selected. Again, this was done twice for two parents.

Round 3
The third round was identical to Round 2, except 10% of the population was chosen at random instead of 20%.

Crossovers

The selection of one child from two parents was done in one of the two crossover functions.

Crossover 1
A random number is selected from the range of zero to the length of the individual array. The child is a carbon copy of the first parent from index zero to the index of that random number. The rest of the child array is made from the numbers in the second parent that do not already exist in the child.

Crossover 2
The procedure is identical to Crossover 1, but in addition, the steps are completed vice versa for the second parent and therefore two child arrays are produced. The fitnesses of both are compared, and the best one is chosen.

Mutations

The mutation of the child was needed in order to introduce more variety in the population "gene pool" and also to increase the randomness. Each child array was mutated before being added to the next generation's population. Mutations 2 and 3 were adapted from TutorialsPoint[1].

Mutation 1
The first mutation function simply swapped two random indices in the child array.

Mutation 2

The second mutation took a random slice of indices in the child array and scrambled it before piecing the array back together.

Mutation 3

The third mutation took a random slice of indices in the child array and reversed them. The idea was that a certain arrangement in the child array was valuable and therefore reversing it might maintain the fitness while introducing an element of randomness.

No Mutation

To test out what would happen if none of the children were mutated, one run of the algorithm was done with each child array added to the new generation without any mutation at all.

Number of Generations

The number of generations refers to how many times a new population was created from new individuals.

50

The default number of generations was fifty. This was maintained throughout most of the experiment.

100

Once the best individuals had been retrieved from the combination of parent selections, crossovers, and mutations, the number of generations was increased to one hundred.

Population Size

The population size refers to how many individuals existed in one generation.

250

The default population size was 250. Like the number of generations, this was maintained throughout the experiment until the best combination of parent selections, crossovers, and mutations was found.

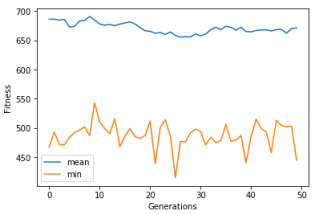
350

The population size was increased to 350 while keeping the number of generations at fifty.

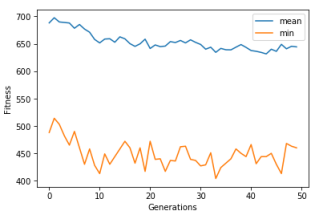
Fifteen Cities

The results of the various combinations for the dataset containing information for fifteen cities is below. The captions contain the best individual and that individual's fitness.

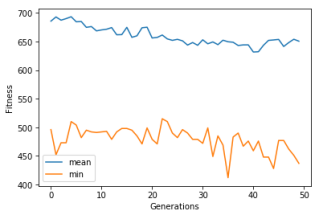
Round 1, Crossover 1, Generations: 50, Population: 250



(a) Mutation 1, best individual: 8 4 12 0 1 10 3 5 6 7 9 2 11 13 14, fitness: 445



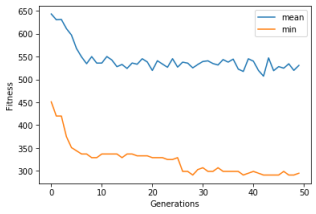
(b) Mutation 2, best individual: 3 5 13 11 10 0 1 2 7 12 14 8 4 6 9, fitness: 460



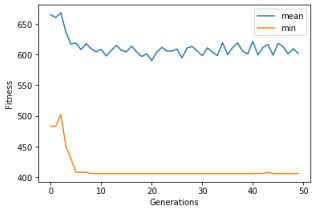
(c) Mutation 3, best individual: 1 0 10 3 5 13 2 4 6 7 8 9 11 14 12, fitness: 437

As shown here, none of the algorithms have converged, although they are gently trending down.

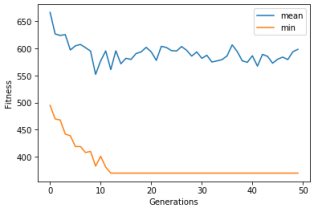
Round 2, Crossover 1, Generations: 50, Population: 250



(a) Mutation 1, best individual: 13 9 7 5 3 10 0 12 1 14 8 4 6 2 11, fitness: 291



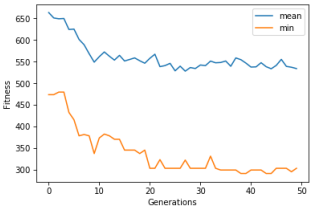
(b) Mutation 2, best individual: 0 10 3 7 13 9 4 6 2 11 5 8 14 1 12, fitness: 406.0



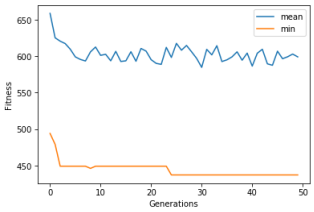
(c) Mutation 3, best individual: 0 1 11 13 2 6 4 14 8 9 7 5 3 10 12, fitness: 370

These algorithms, on the other hand, have converged quickly. However, Figure a produced the best individual.

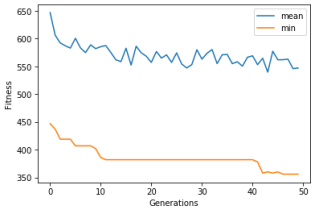
Round 3, Crossover 1, Generations: 50, Population: 250



(a) Mutation 1, best individual: 2 6 4 14 8 1 12 0 10 3 5 7 9 11 13, fitness: 303



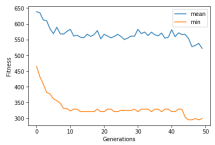
(b) Mutation 2, best individual: 1 0 3 5 7 2 11 13 14 8 4 6 9 10 12, fitness: 437



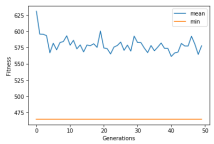
(c) Mutation 3, best individual: 11 9 2 6 4 8 14 12 1 0 3 10 5 7 13, fitness: 356

Extras with Round 2

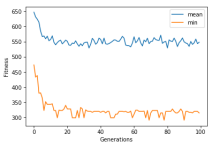
Because Round 2 produced the best results, some extra algorithms were run with that parent selection function.



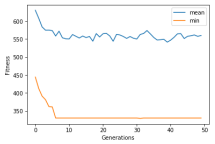
(a) Crossover 2, Mutation 1, Gens: 50, Pop: 250, best individual: 2 6 4 8 14 1 12 0 10 3 5 7 9 11 13, fitness: 299



(b) Crossover 1, No mutation, Gens: 50, Pop: 250, best individual: 9 6 4 8 14 1 12 0 2 3 5 7 10 11 13, fitness:465



(c) Crossover 1, Mutation 1, Gens: 100, Pop: 250, best individual: 6 4 8 14 1 12 0 10 3 5 13 7 9 11 2, fitness: 315



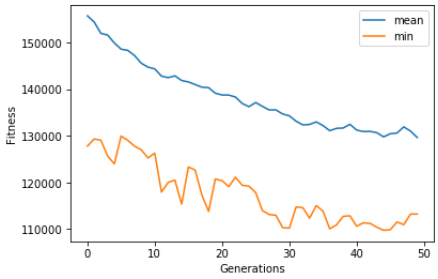
(d) Crossover 1, Mutation 1, Gens: 50, Pop: 350, best individual: 8 14 4 6 2 11 13 1 12 0 10 3 5 7 9, fitness: 330

As shown here, Figure b with no mutation did not perform well at all. Mutation is necessary for the algorithm to function properly.

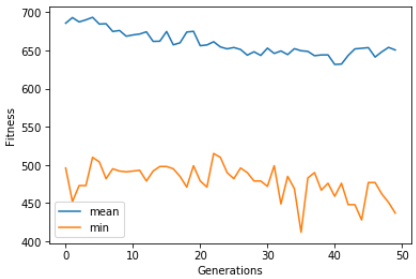
Forty-nine Cities

The results of the fifteen-city algorithms were taken account, so not all combinations were tried with the forty nine-city dataset. Instead, only the best were selected.

Round 1, Crossover 1, Generations: 50, Population: 250



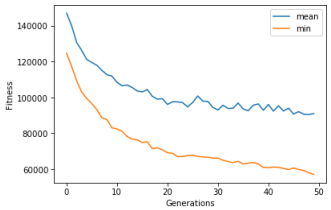
(a) Mutation 1, best individual:34 44 9 2 1 28 3 33 21 40 5 6 7 0 31 10 11 12 13 4 8 14 15 16 17 18 19 20 22 23 24 25 26 27 29 38 32 35 36 37 30 39 41 42 43 45 46 47, fitness: 113246



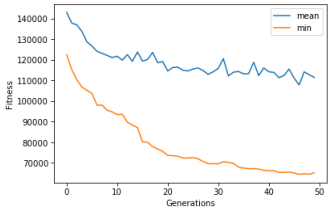
(b) Mutation 3, best individual: 13 38 46 9 4 3 25 1 47 28 12 7 8 10 11 14 15 0 2 5 6 16 17 18 19 20 21 22 23 24 26 27 29 30 31 32 33 34 35 36 37 39 40 41 42 43 45 44, fitness: 110229

Neither of these have converged.

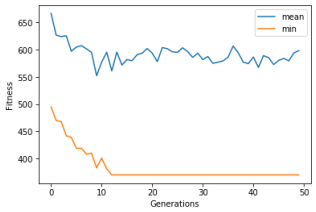
Round 2, Generations: 50, Population: 250



(a) Mutation 1, Crossover 1, best individual: 31 44 34 9 3 25 1 41 28 2 40 10 24 13 38 23 47 4 21 0 7 15 33 22 37 8 39 12 20 46 19 16 42 5 6 11 14 30 18 26 17 27 29 36 35 32 43 45, fitness: 57001



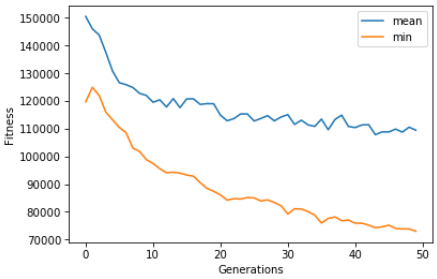
(b) Crossover 2, Mutation 1, best individual is: 4 13 28 40 15 37 30 39 8 0 21 2 33 1 41 23 22 19 27 36 5 29 31 47 34 44 25 3 9 38 10 11 14 32 20 24 12 7 42 6 17 16 26 18 35 43 45 46, fitness: 65183



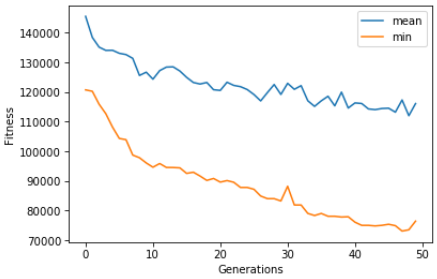
(c) Crossover 1, Mutation 3, best individual: 27 18 26 42 16 17 43 37 0 8 14 38 1 21 35 7 22 47 4 44 3 31 46 32 5 36 19 45 39 10 11 33 15 40 28 25 34 41 9 23 2 6 30 12 13 20 24 29, fitness: 79789

Only Figure c has converged, but it has not reached the best fitness. Figure a produces the best individual for the forty-nine cities problem.

Round 3, Crossover 1, Generations: 50, Population: 250



(a) Mutation 1, best individual is: 33 40 21 0 22 38 28 1 10 39 14 24 13 44 34 3 25 41 47 4 31 9 23 20 12 15 2 37 8 7 6 16 5 11 17 18 19 26 27 29 30 32 35 36 42 43 45 46, fitness: 73028



(b) Mutation 3, best individual: 39 2 24 19 41 44 1 9 23 25 12 28 40 10 11 14 37 43 18 16 26 42 17 35 20 13 4 47 38 31 46 21 15 8 7 0 22 3 34 33 32 30 45 36 6 29 27 5, fitness: 76455

Neither of these have converged, and the fitnesses are significantly higher than those above.

Conclusions

The best solution for the fifteen-city problem is sequence of 13 9 7 5 3 10 0 12 1 14 8 4 6 2 11 with a total distance travelled of 291.

The best solution for the forty nine-city problem is sequence 31 44 34 9 3 25 1 41 28 2 40 10 24 13 38 23 47 4 21 0 7 15 33 22 37 8 39 12 20 46 19 16 42 5 6 11 14 30 18 26 17 27 29 36 35 32 43 45 with a total distance travelled of 57,001.

Throughout this experiment, it was found that randomness was valued and produced better results than non-random functions. For example, Crossover 2 was not successful. Crossover 2 created two child arrays which were then compared for fitness, and the lowest one was selected. Although it was nearly identical to Crossover 1, it produced far worse results.

Mutation 2 was also found to be ineffective. This involved taking a slice of the child array at a random index of a random size and scrambling it. This was perhaps too much randomness and produced the worst results of all three mutations.

The combination that produced the best results for both the fifteen-city problem and the forty nine-city problem was:

- Selecting the parents by randomly choosing 20% of the population and choosing the best individual among that selection
- Randomly selecting a number, making the child array a carbon copy of the first parent from index zero to the index of that random number, and then filling in the rest of the child from missing numbers from the second parent
- Randomly swapping two indices in the child array before adding it to the new population
- Having the number of child arrays in the generation set to 250
- Having the number of generations set to 50

References

1. https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_mutation.htm

2. Dorigo, M. and Gambardella, L.M., 1997. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on evolutionary computation, 1(1), pp.53-66.