**CS 776: Evolutionary Computing -Assignment 4 Fall 2018**

**Full Name:** Batyr Charyyev

**Email address:** [bcharyyev@nevada.unr.edu](mailto:bcharyyev@nevada.unr.edu)

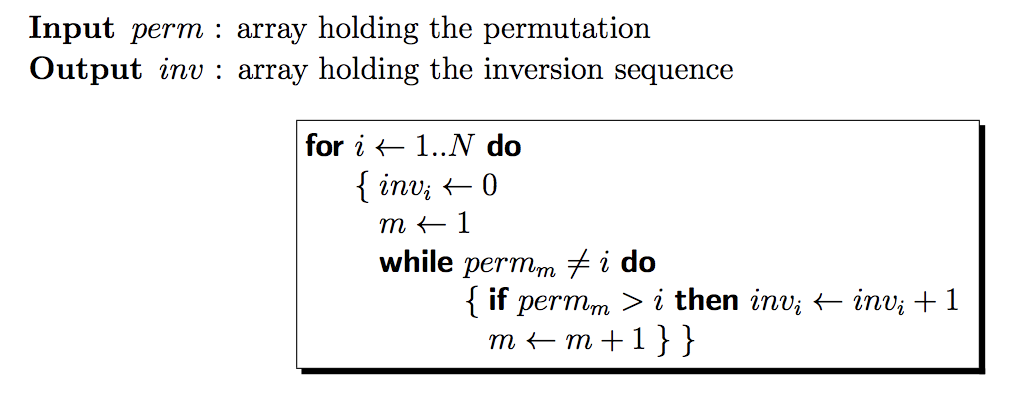
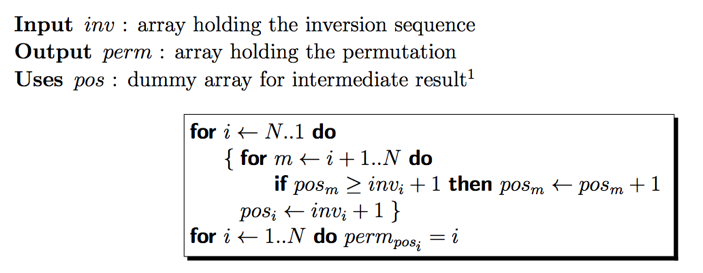
For this assignment I used GA code [1] (implemented in C) of professor Sushil. I modified selection procedure, crossover and mutation procedure of this GA code.

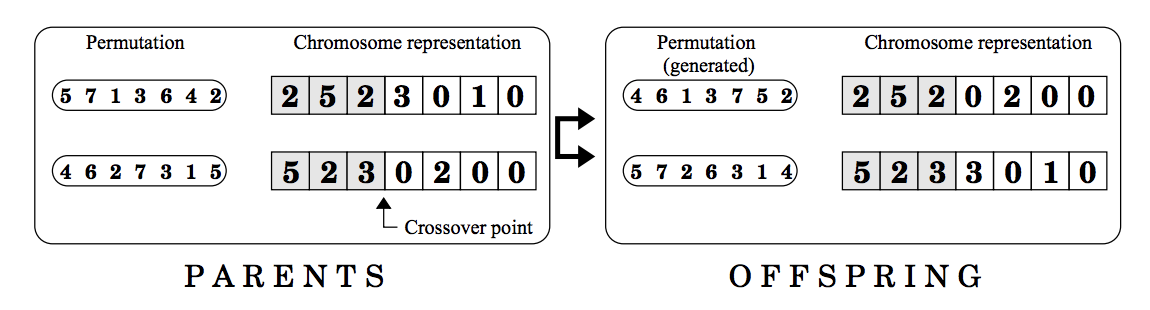
1. **Selection procedure**

I used Elitist selection ((µ + λ) selection) as it was recommended in lecture. By creating *n* children from *n* parents, sort and select best *n* individuals from *2n(parents+children)* individuals. Different from canonical GA, selection procedure keeps best individuals from previous population, by doing this we are able to keep best individuals that we encountered so far.

1. **Crossover**

For cross over I used technique proposed by Gokturk [2].In this paper it first converts permutation of cities into inversion sequence. Then makes crossoever on that inversion sequence and converts it back to permutation of cities. Below I provided algorithm to convert permutation of cities into inversion sequence and inversion sequence back to permutation of cities, also I provided illustration of obtaining offsprings from parents on simple example. All these figures were taken from the article [2].

****

****

1. **Mutation**

For mutation I used Inversion Mutation [3].

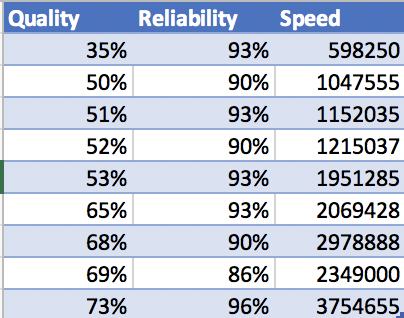
0 **1 2 3 4** 5 6 7

becomes

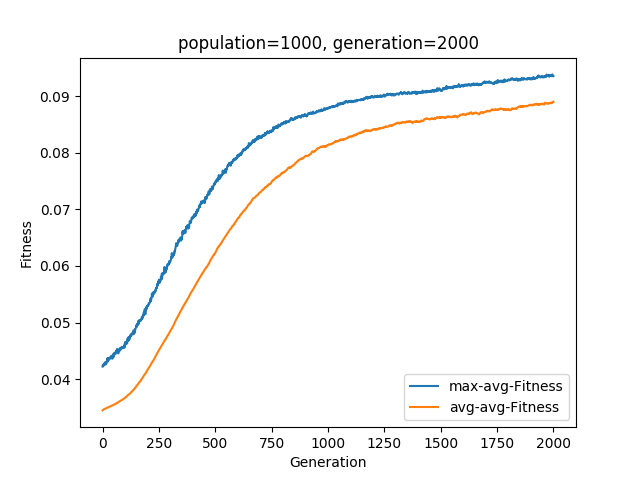
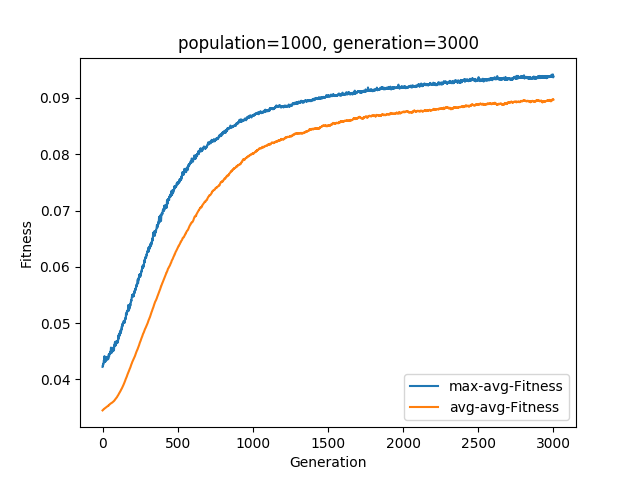
0 **4 3 2 1** 5 6 7

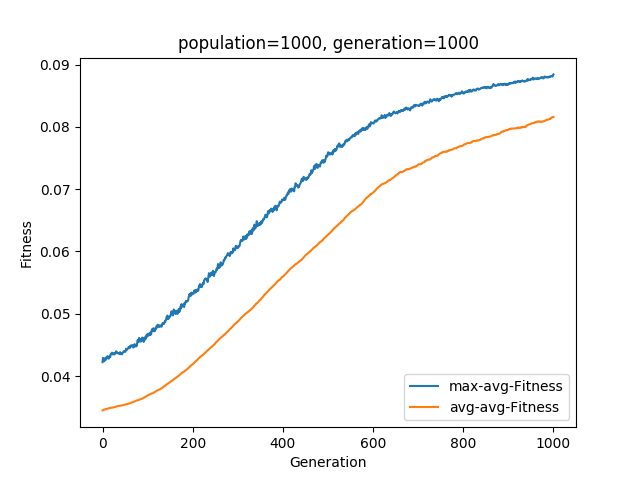
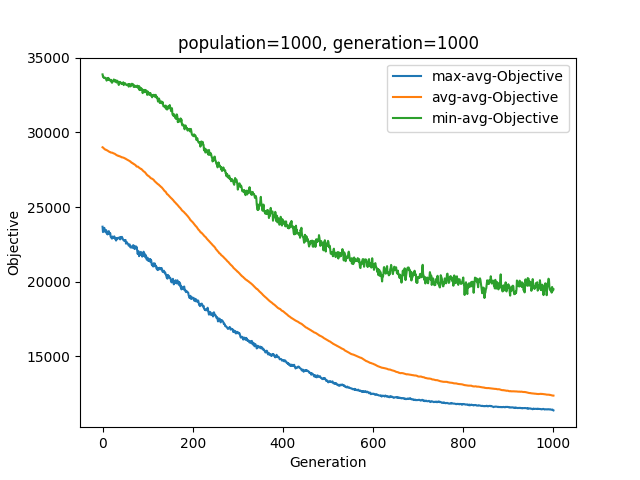
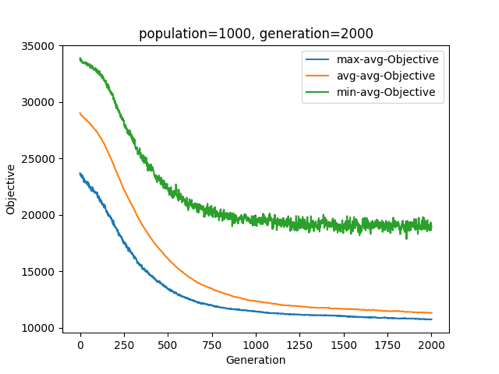
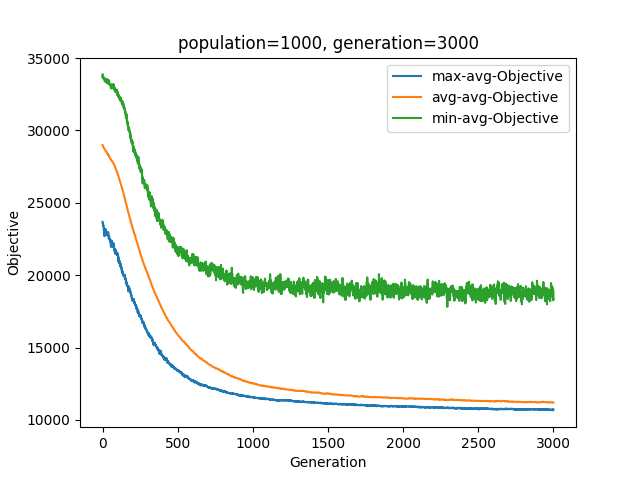
For GA, I took inspiration from [4], this tutorial contains code written in java but I did not use it. Basically each gene represents one city and chromosome represents route that we should take. So in eval function we compute sum of distances between each consecutive cities and try to minimize overall distance.

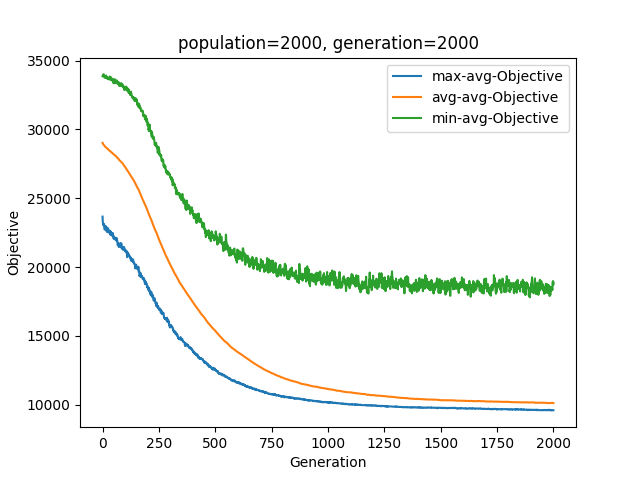
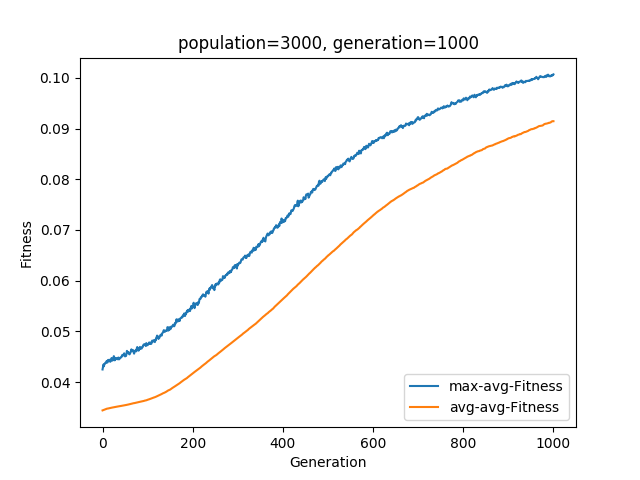
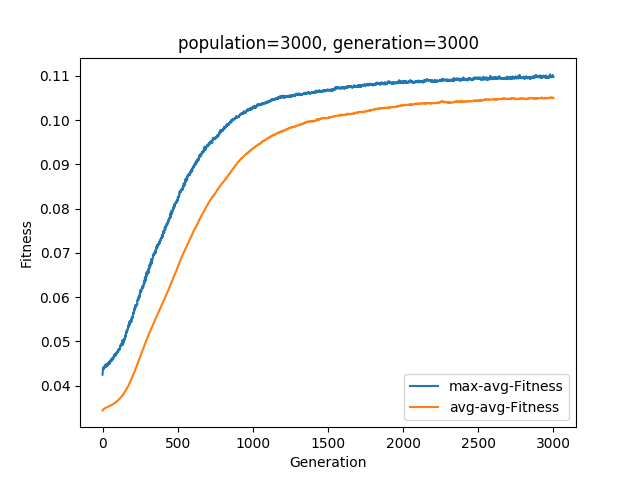
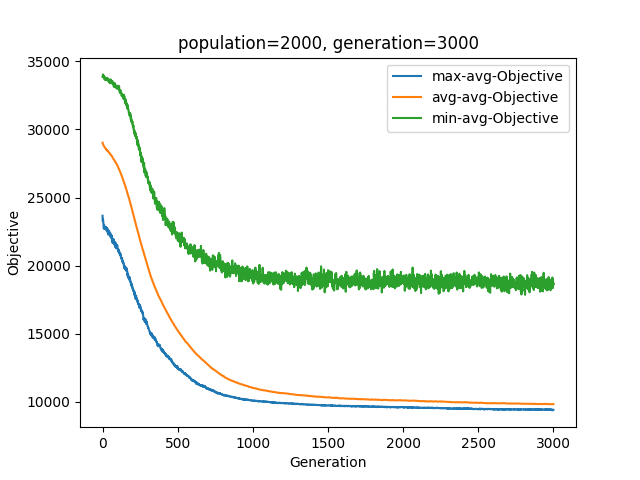
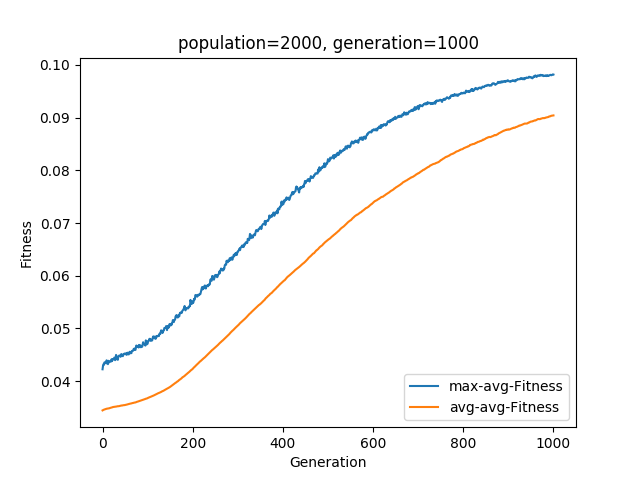
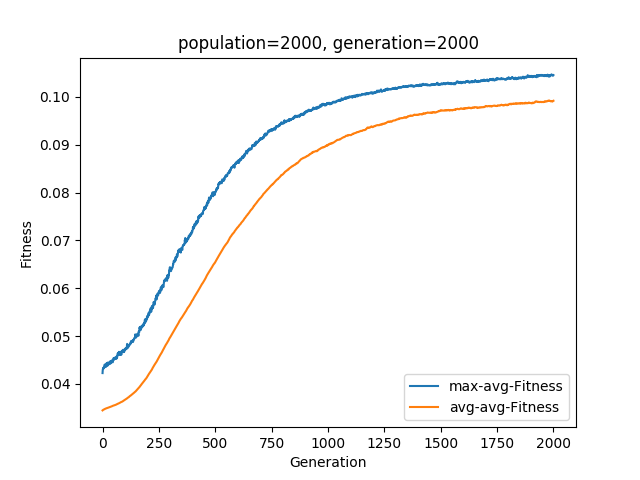
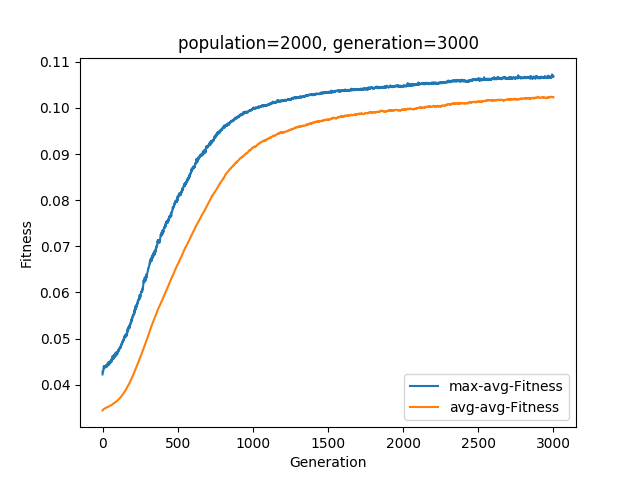
**RESULTS**

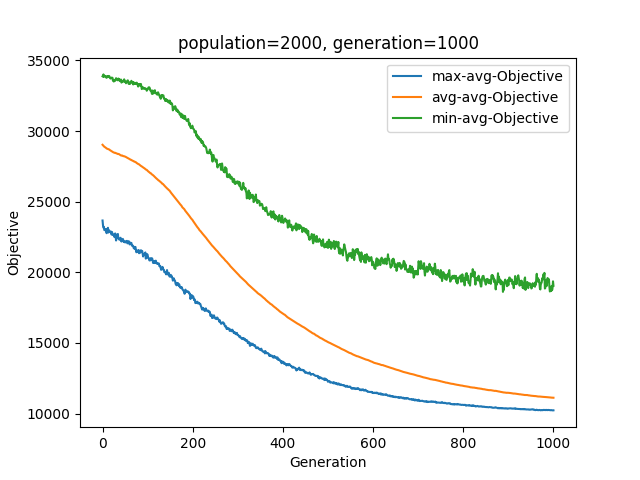
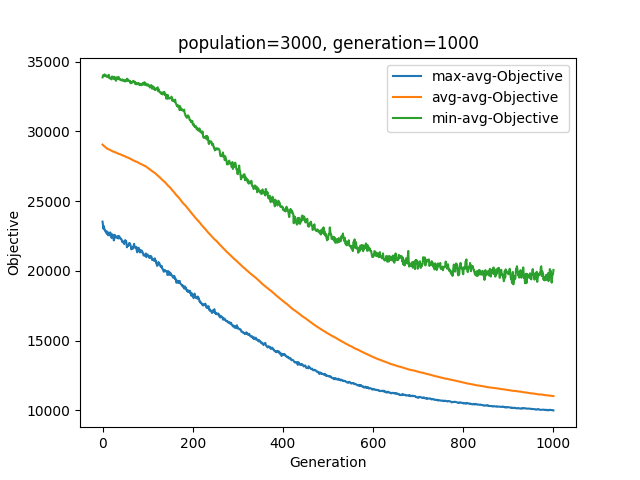
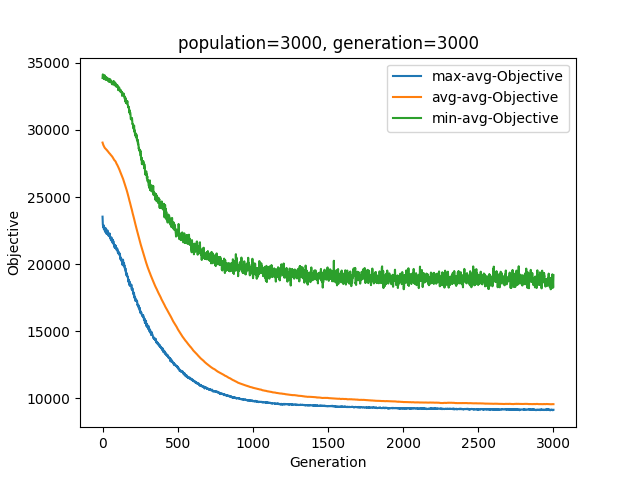


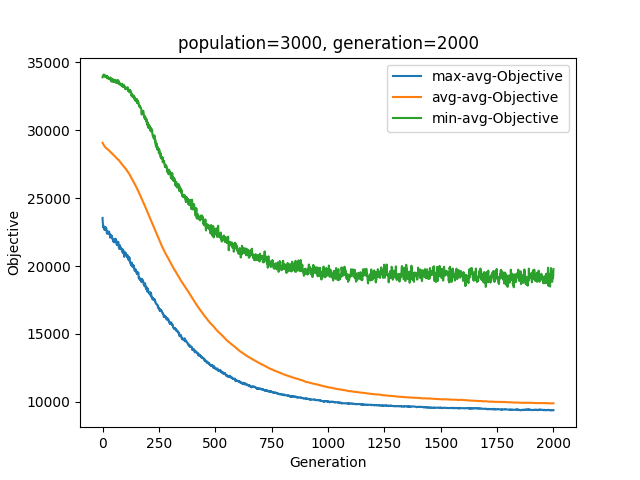
**Berlin52**

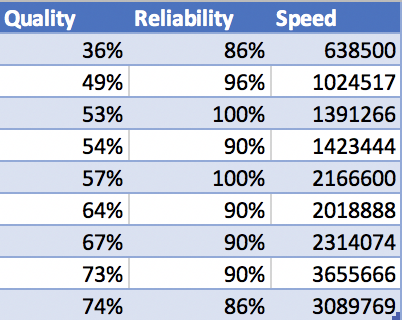


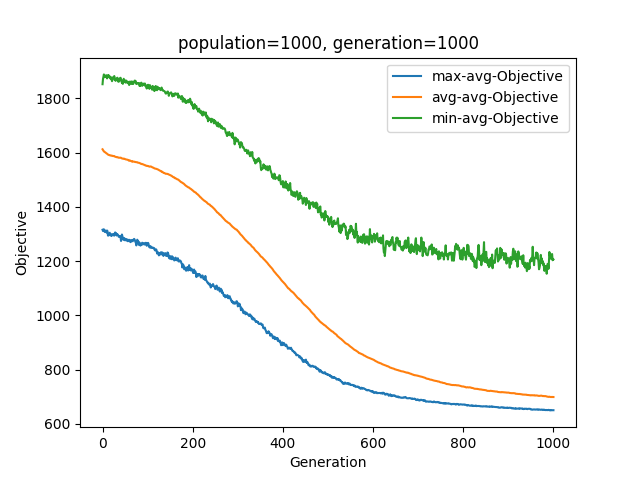
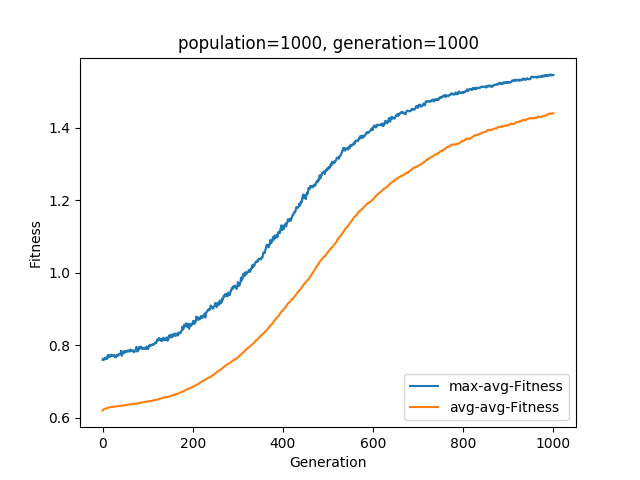
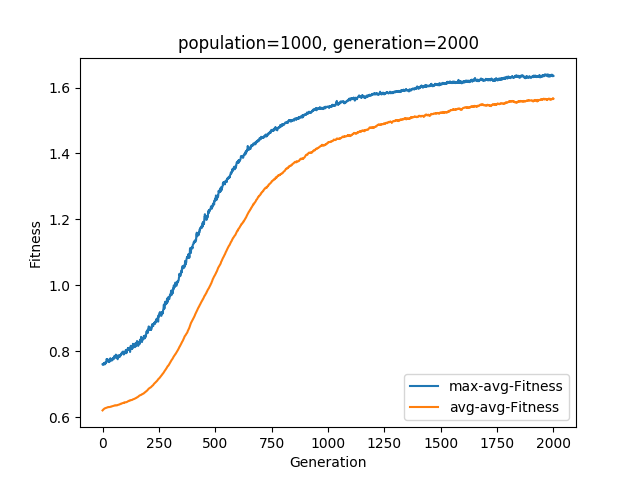


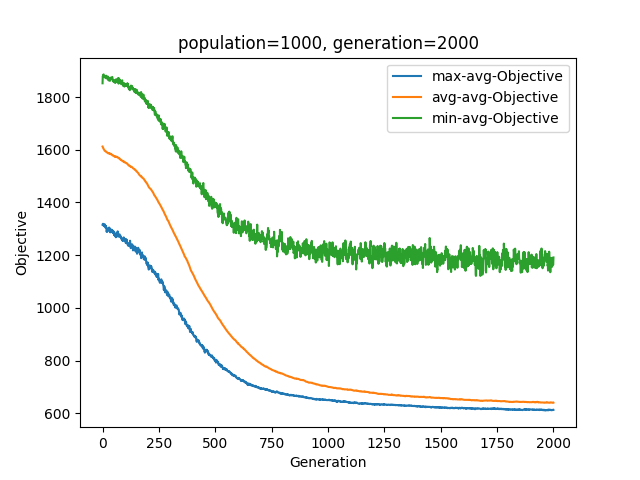
****

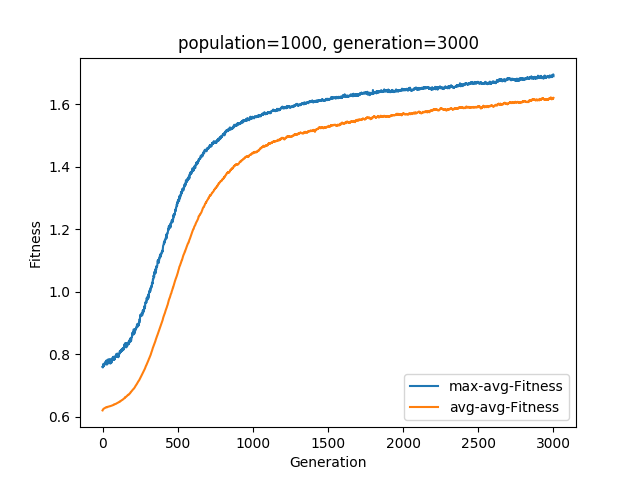


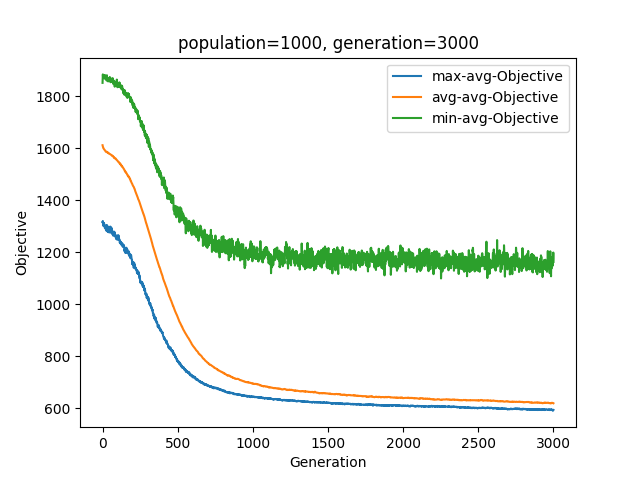


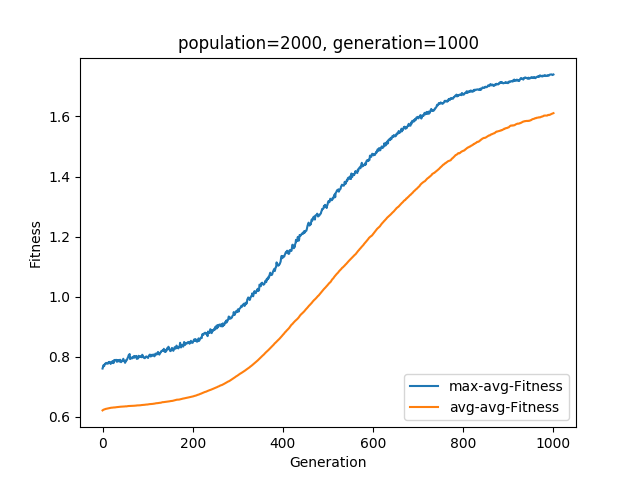
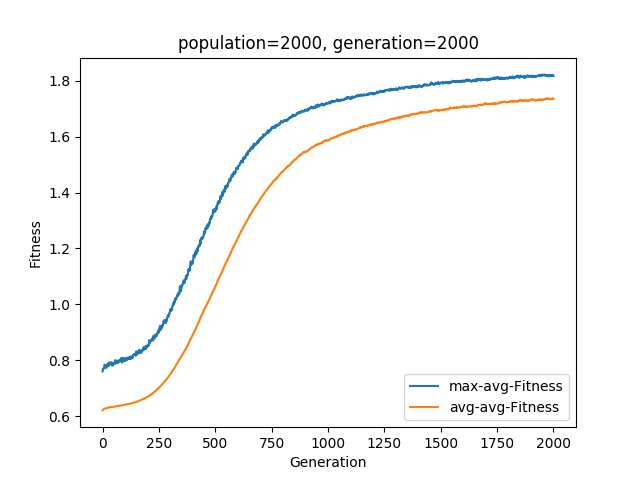
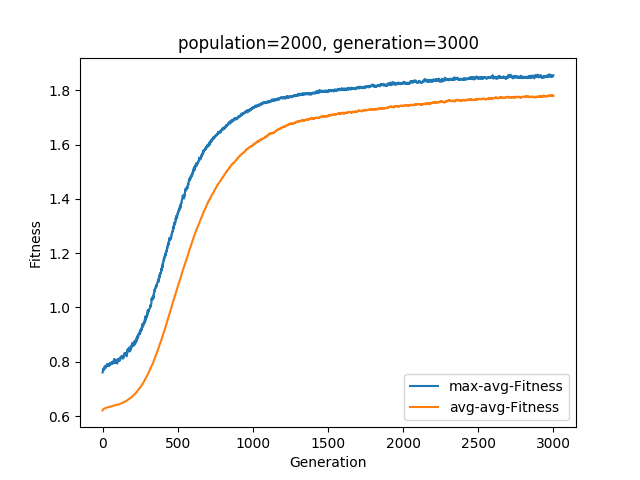
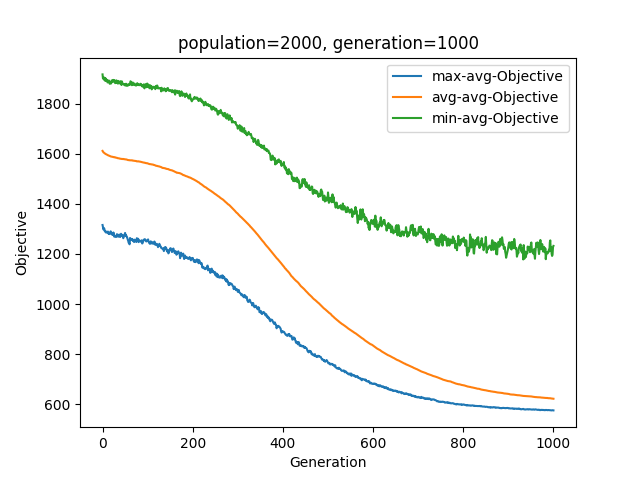
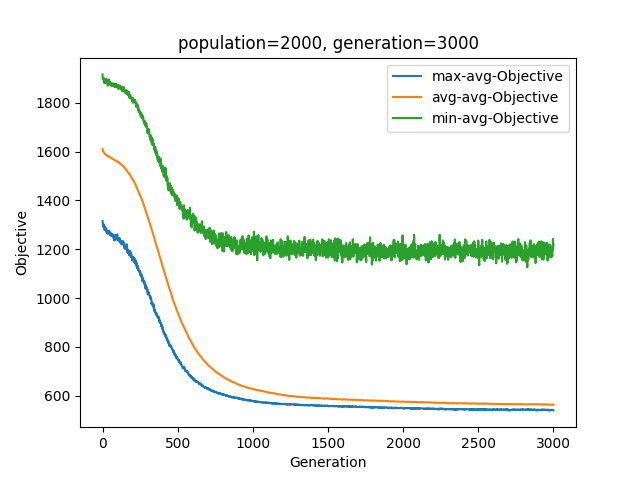
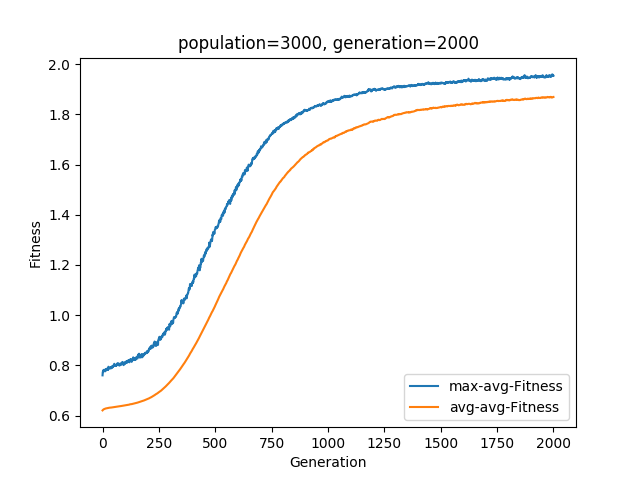
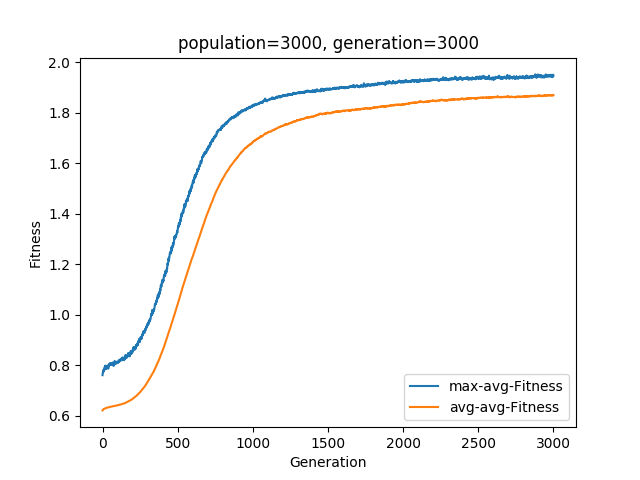
**Eil51**

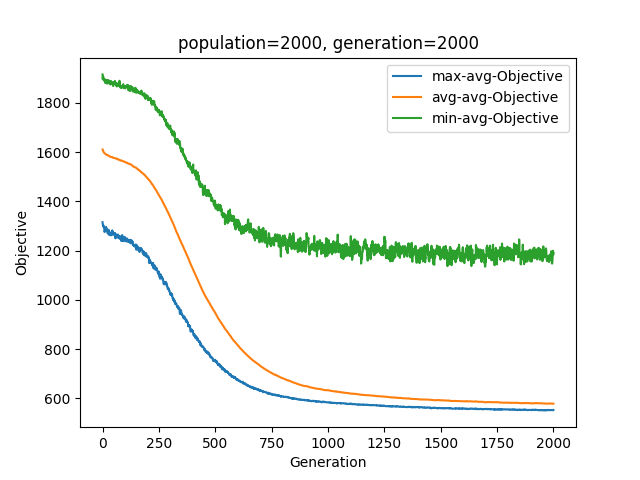
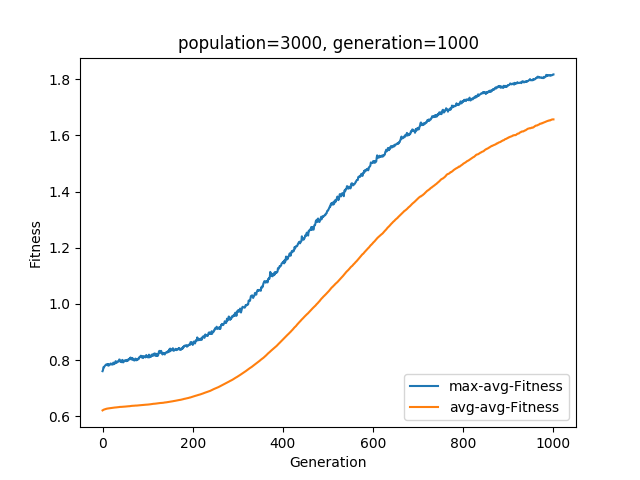


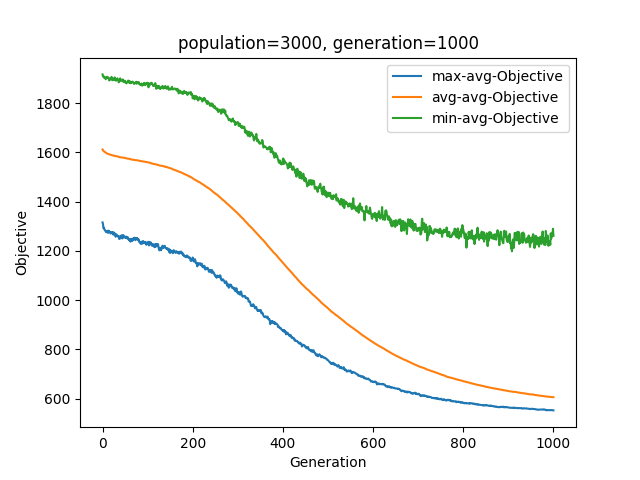
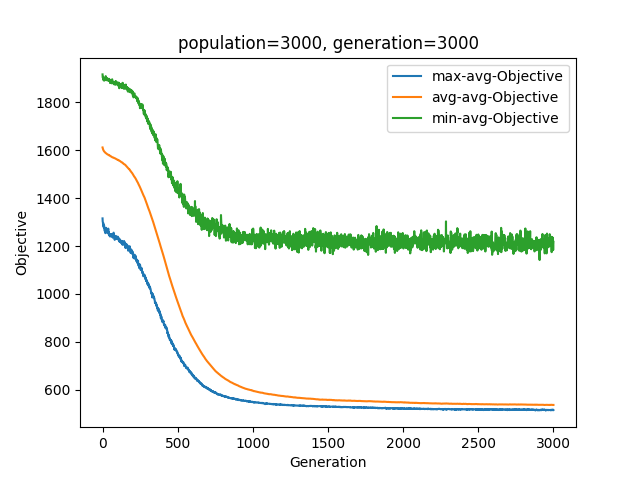
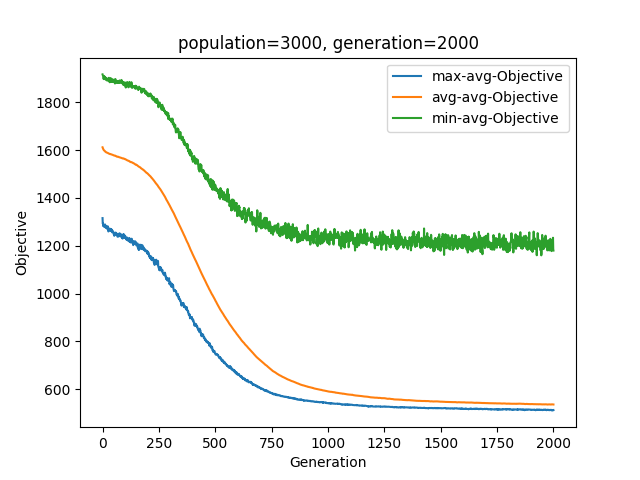




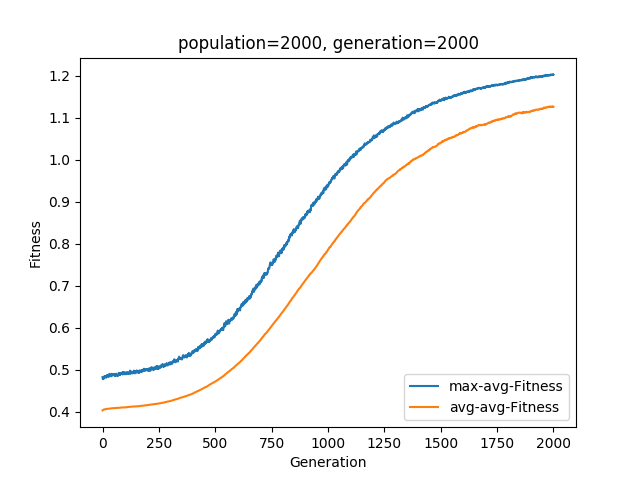


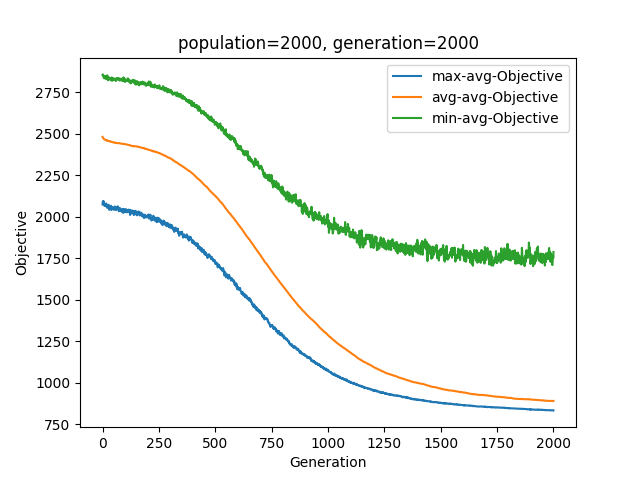


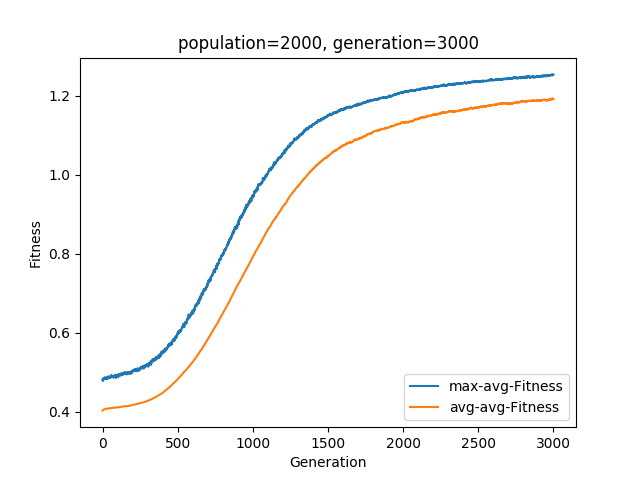


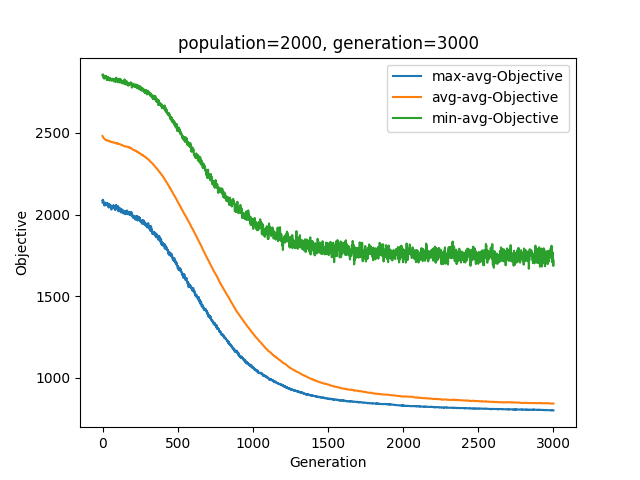


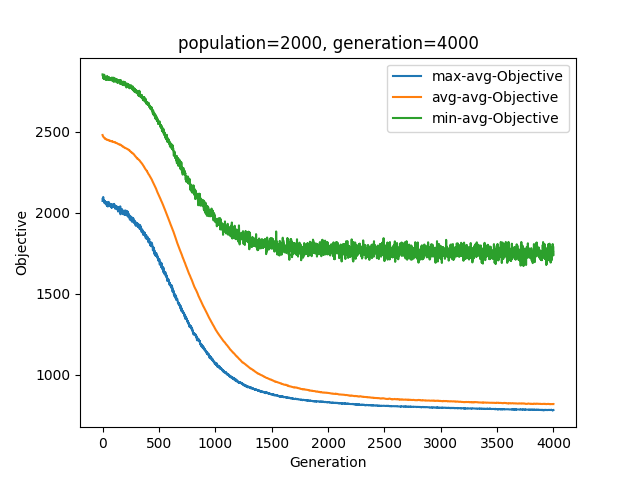
**Eil76**

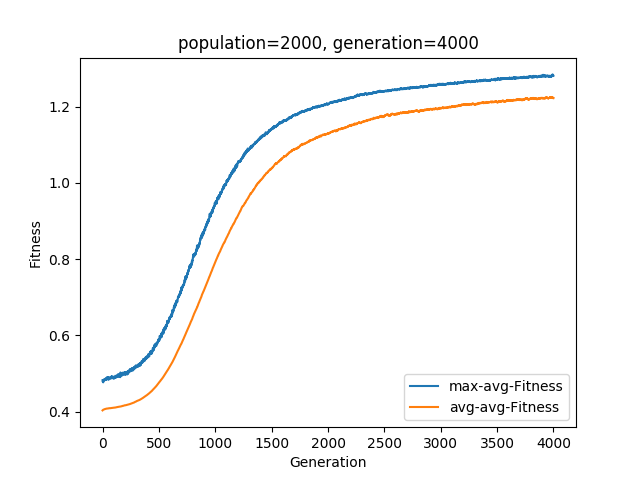


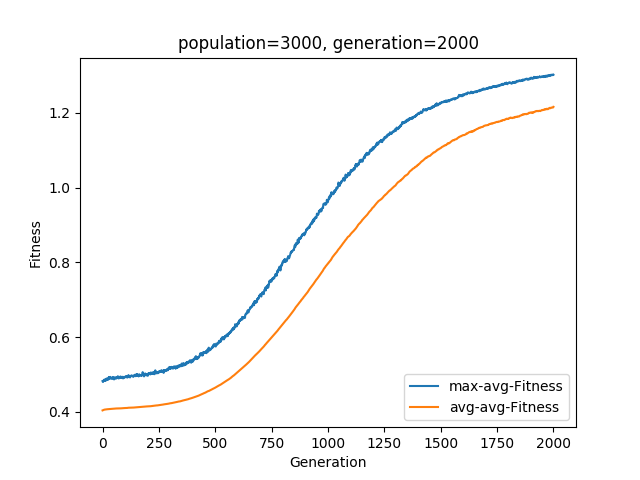
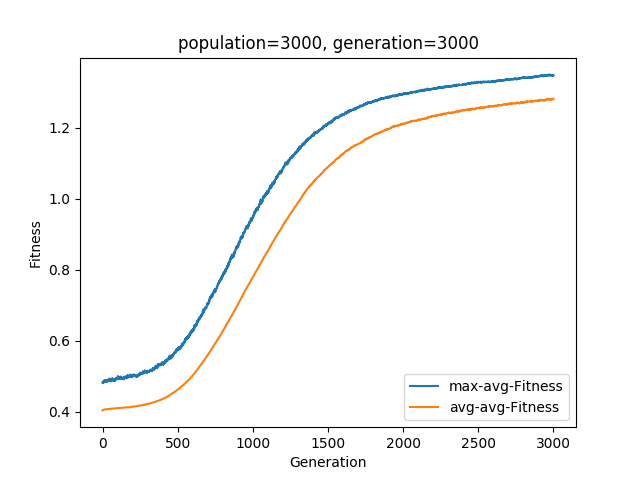
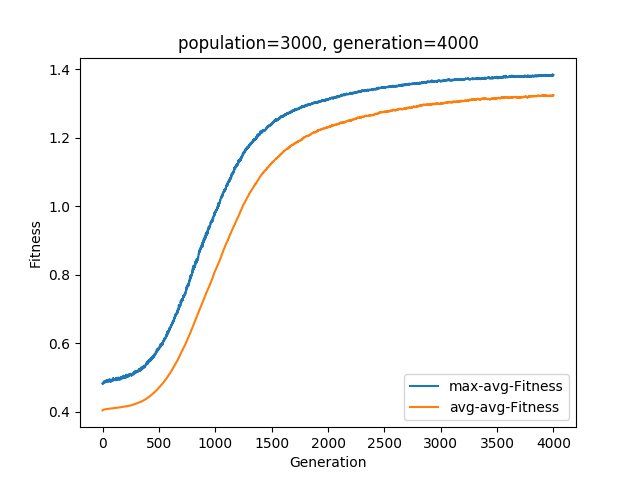
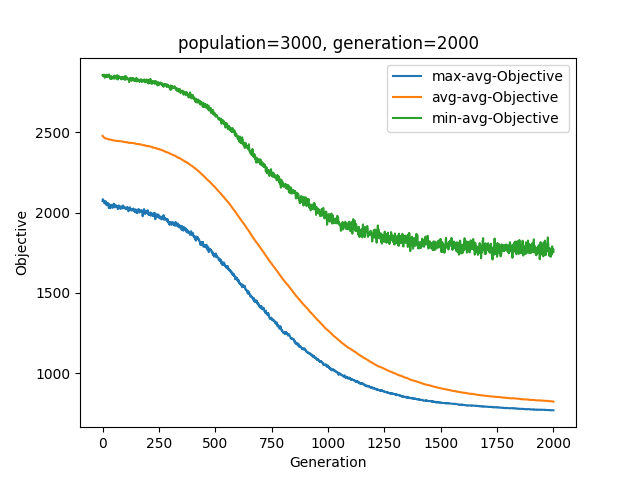
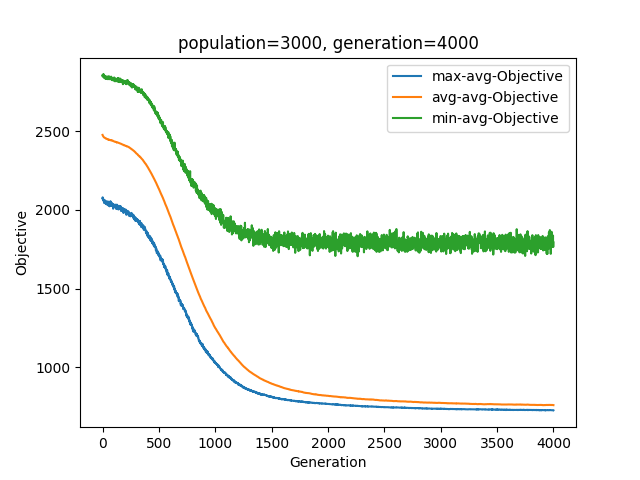
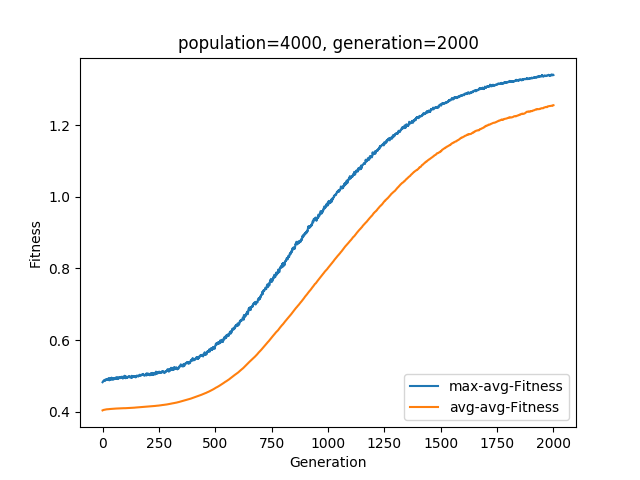
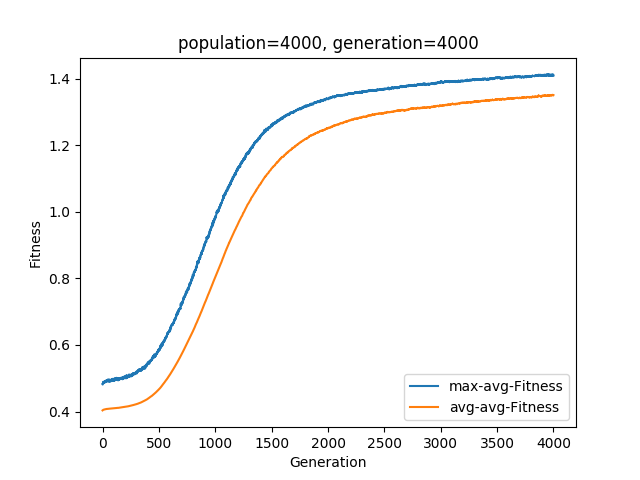
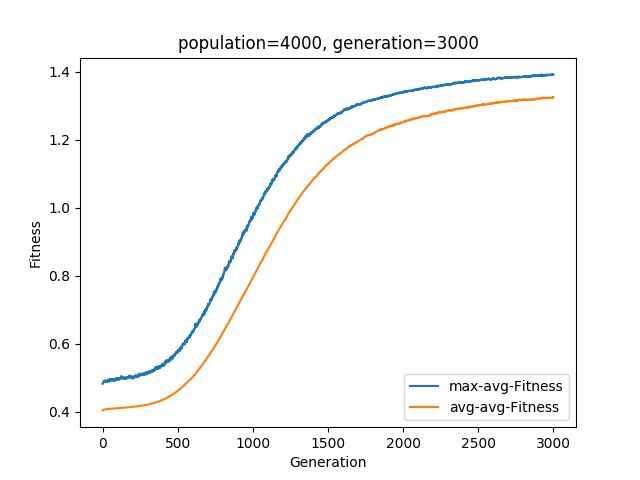


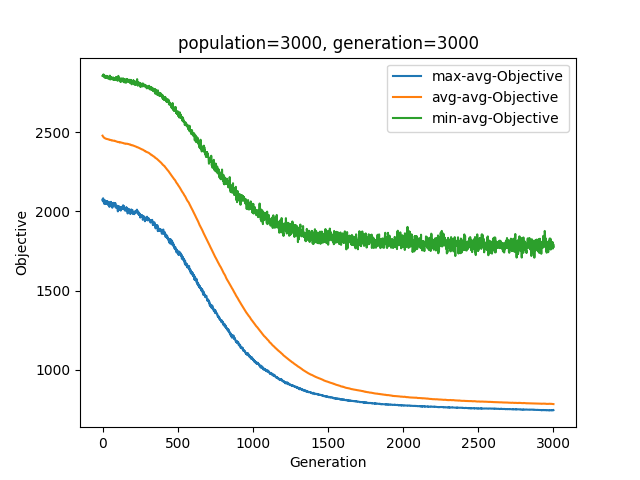
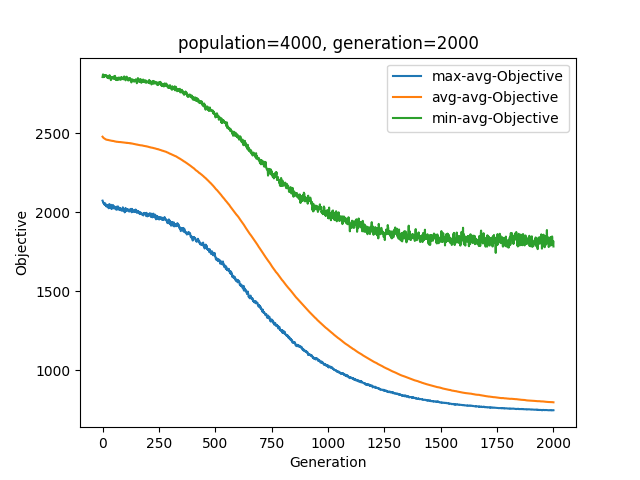
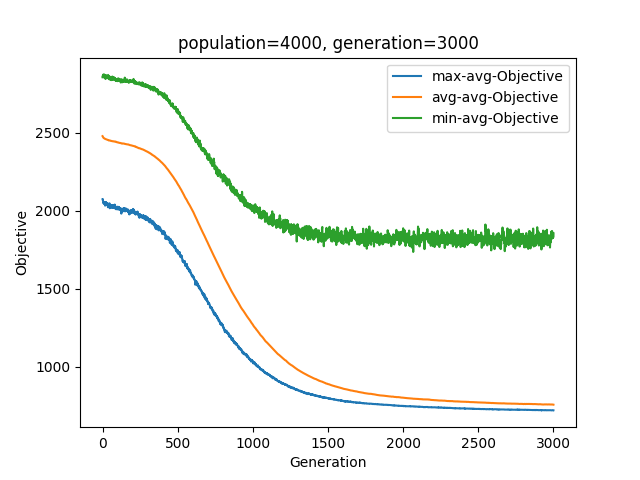
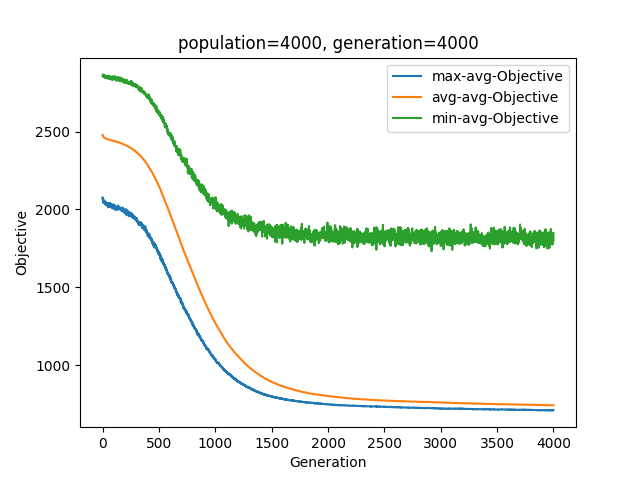


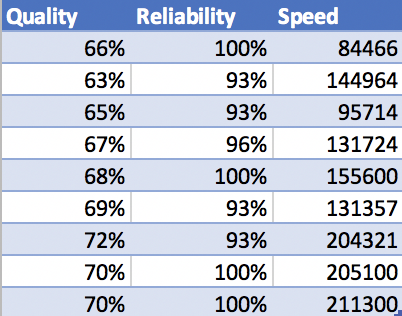


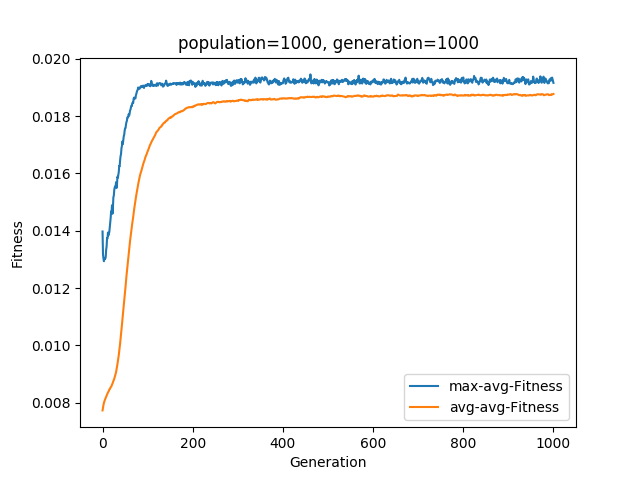
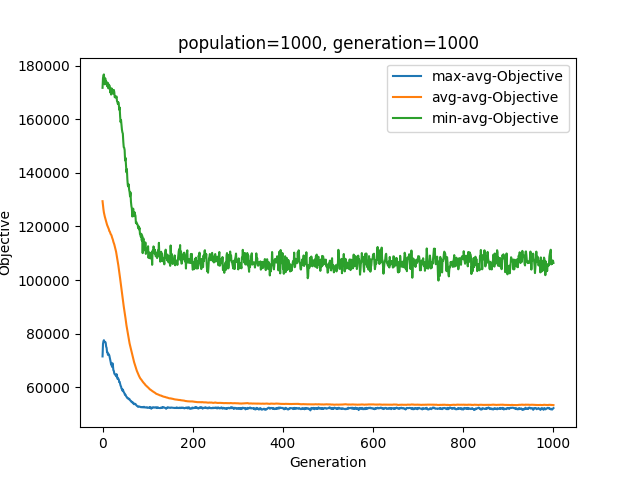


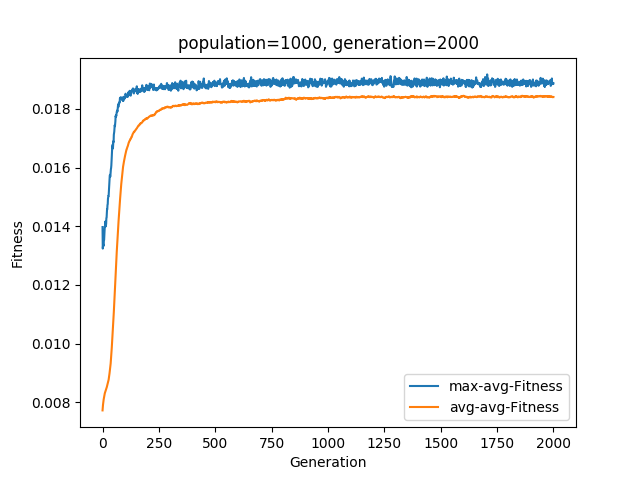
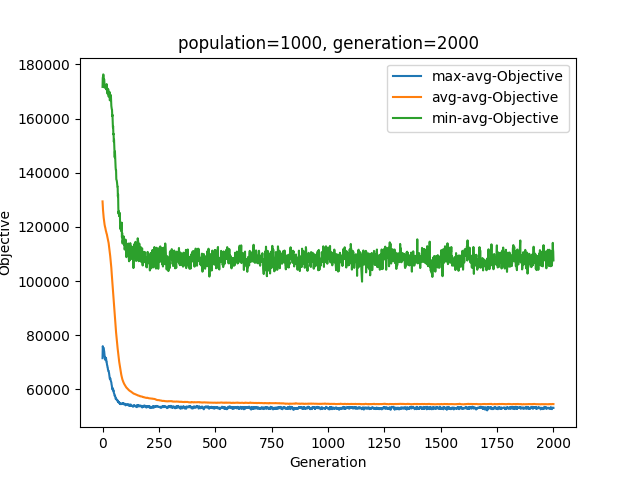


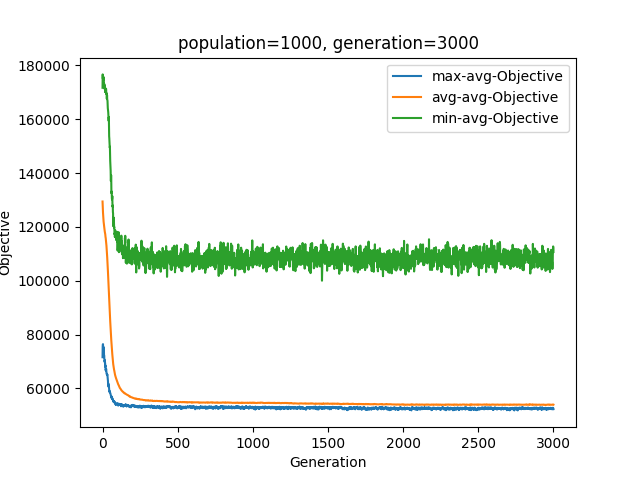
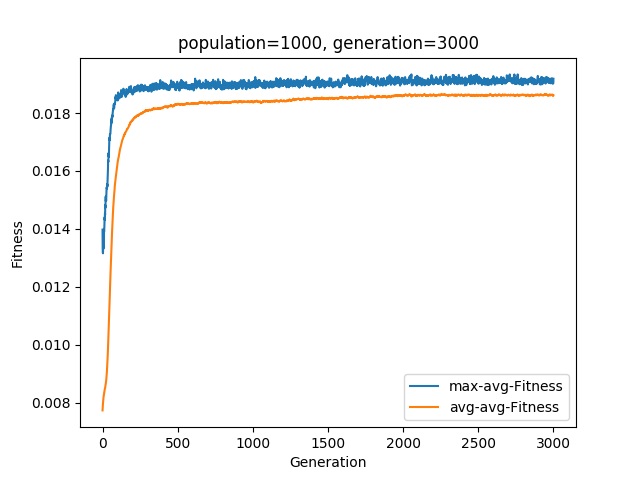


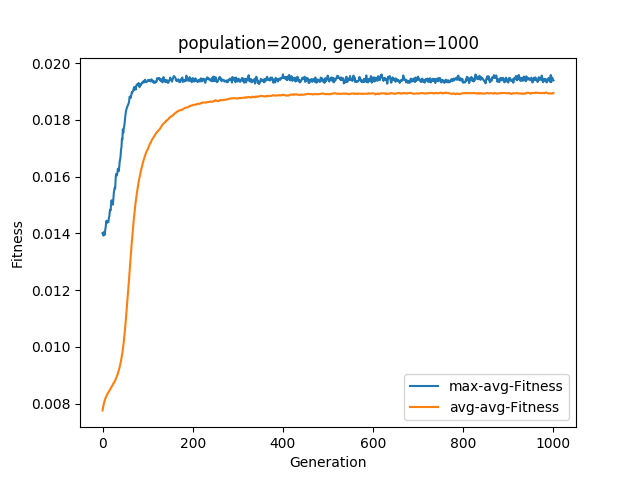
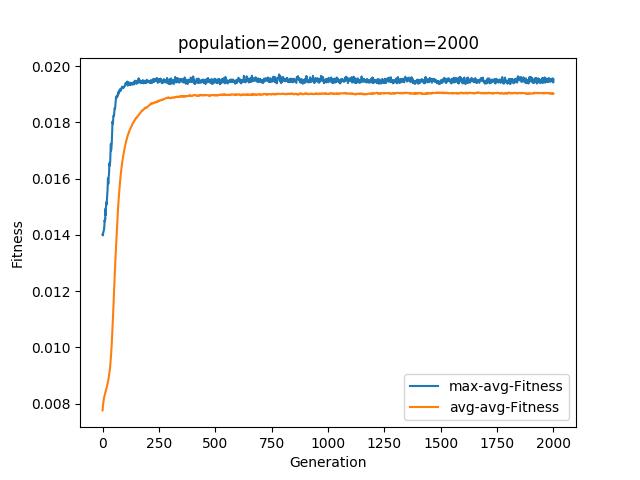
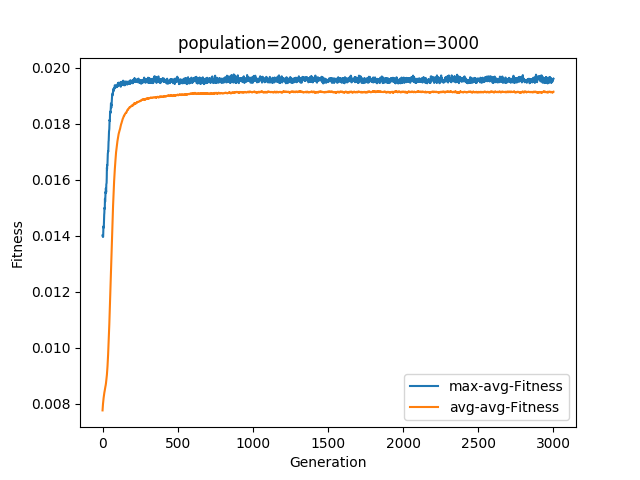
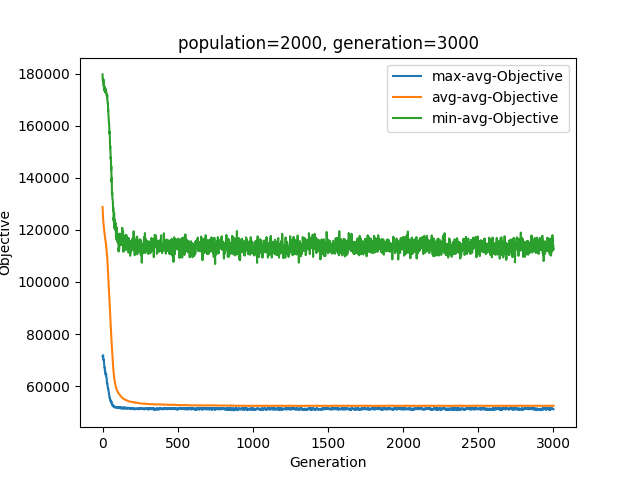


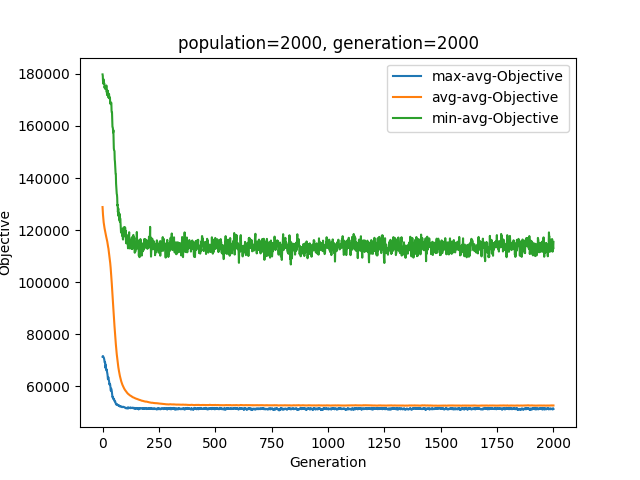
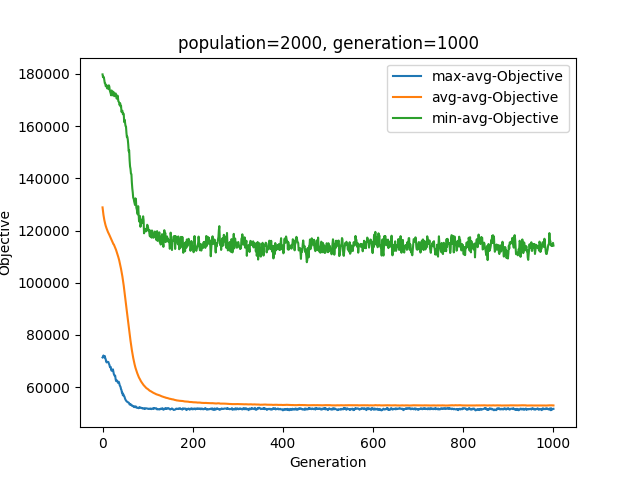
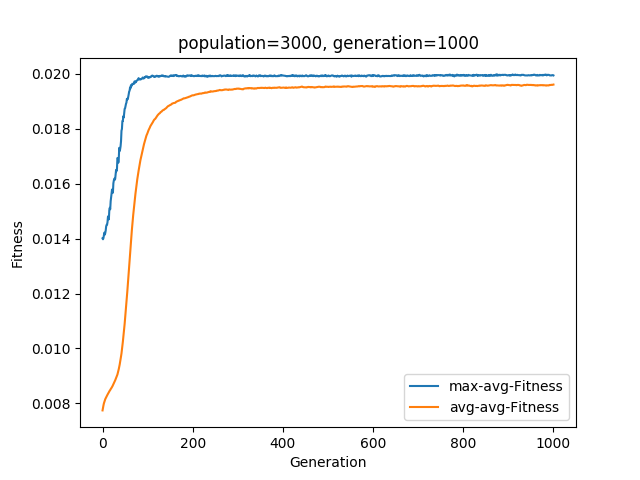
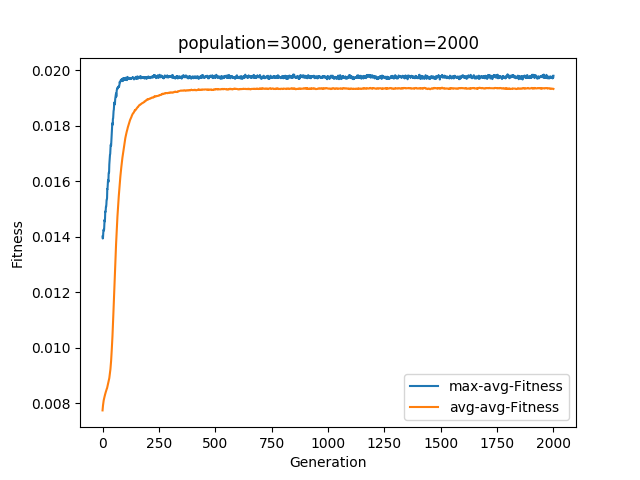
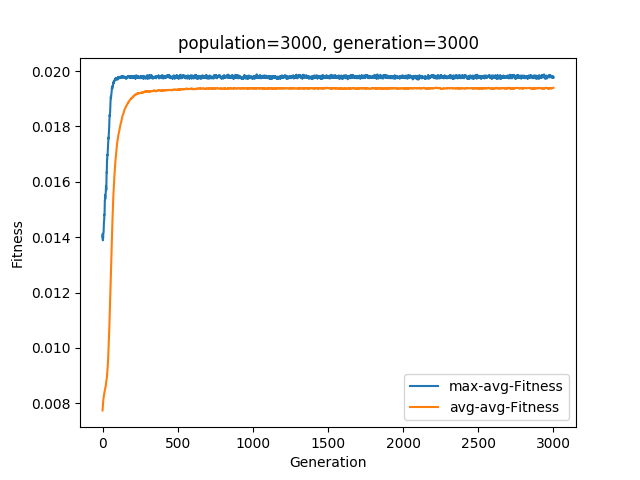
**Burma14**

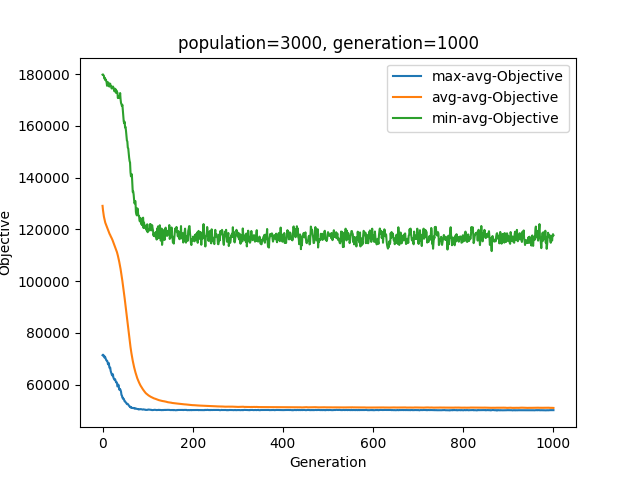
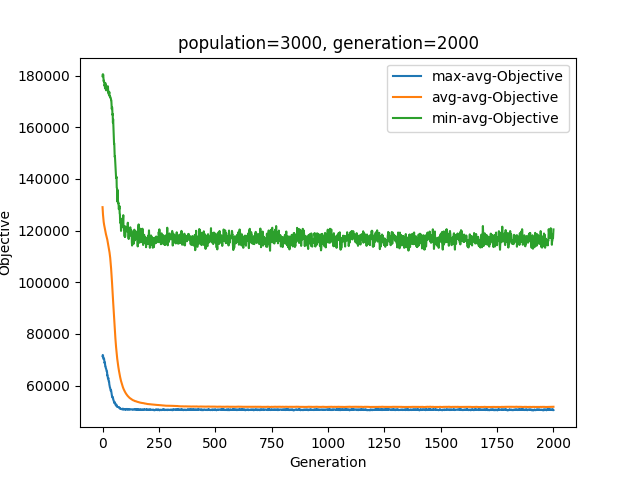




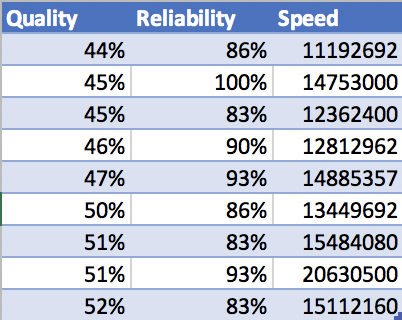


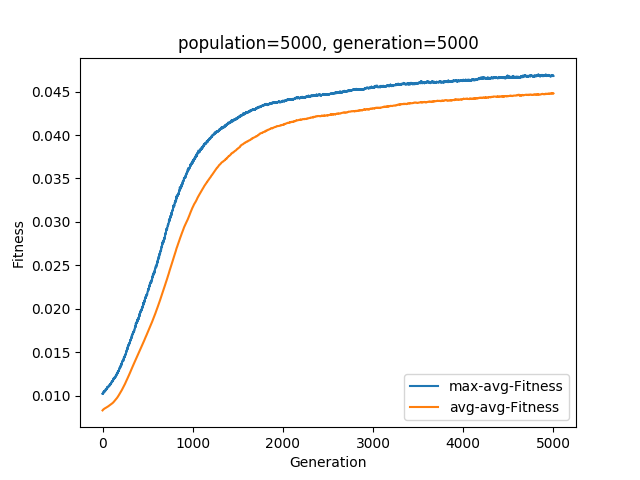


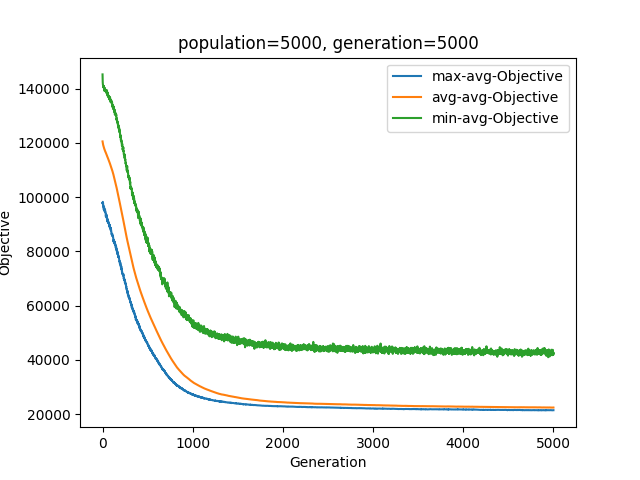


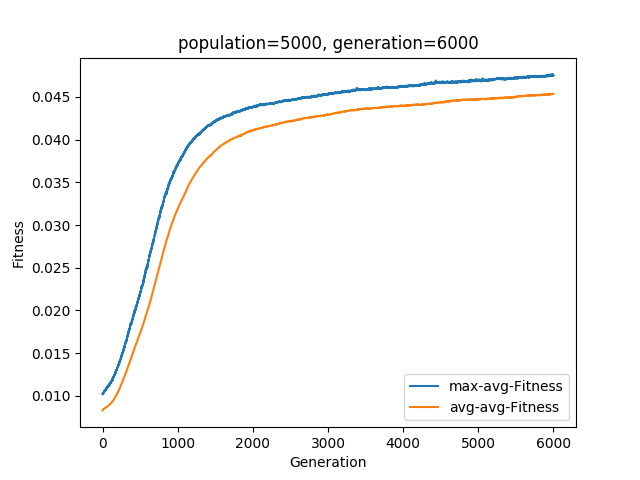
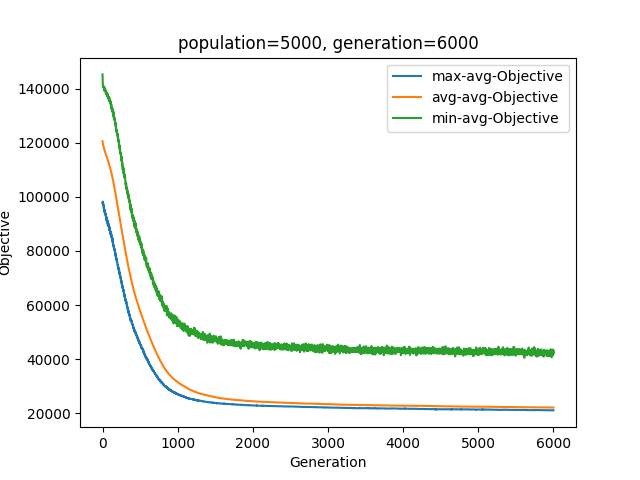


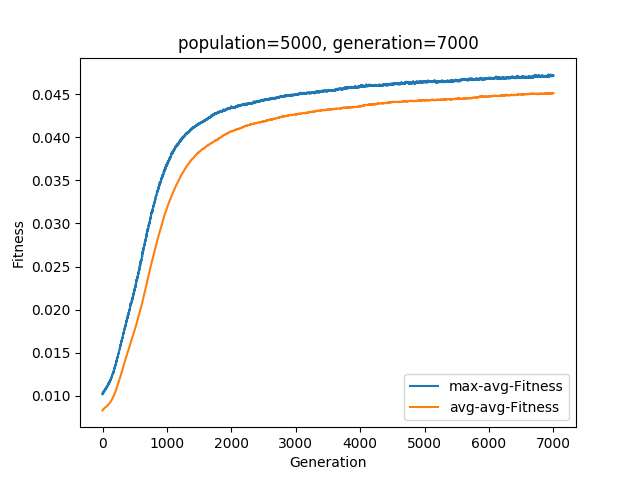
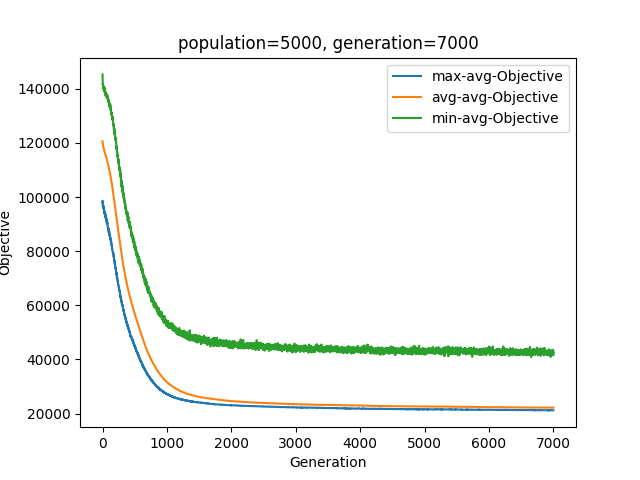


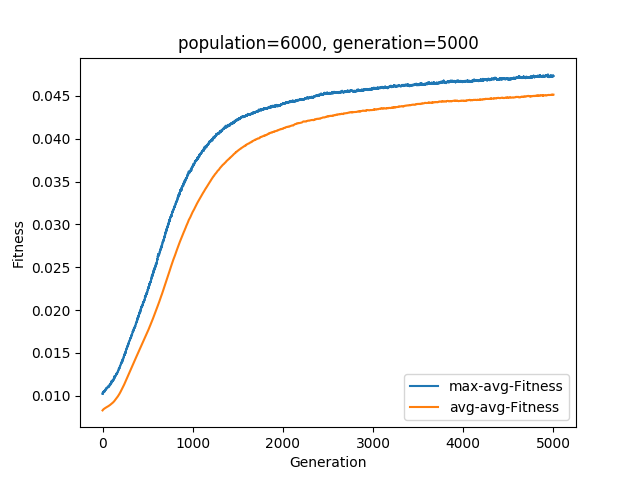
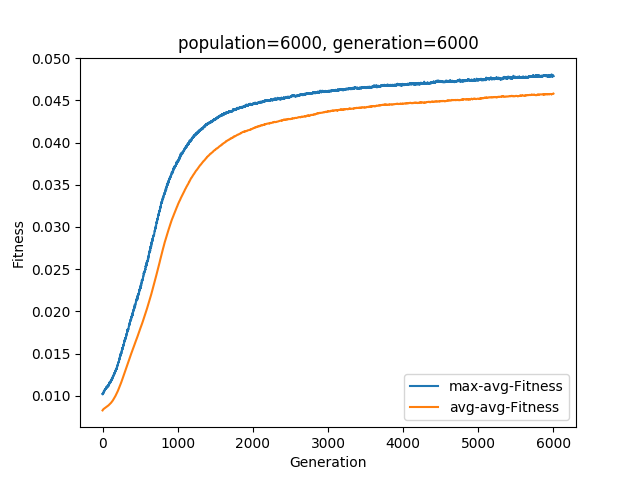
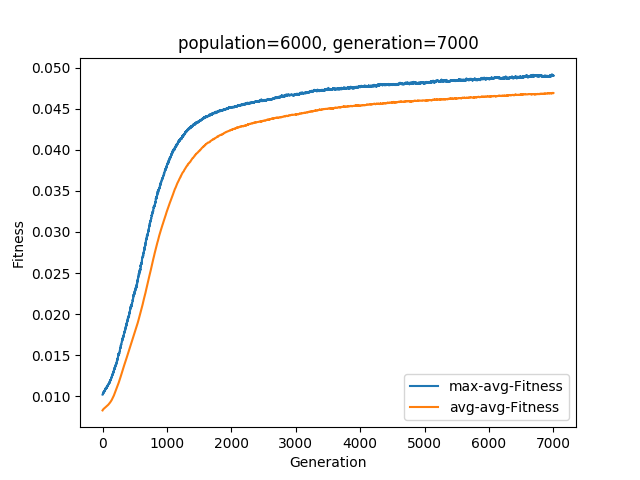
**Lin105**

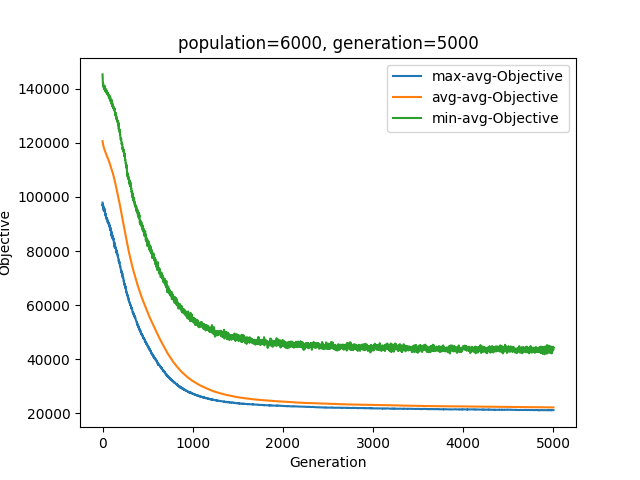
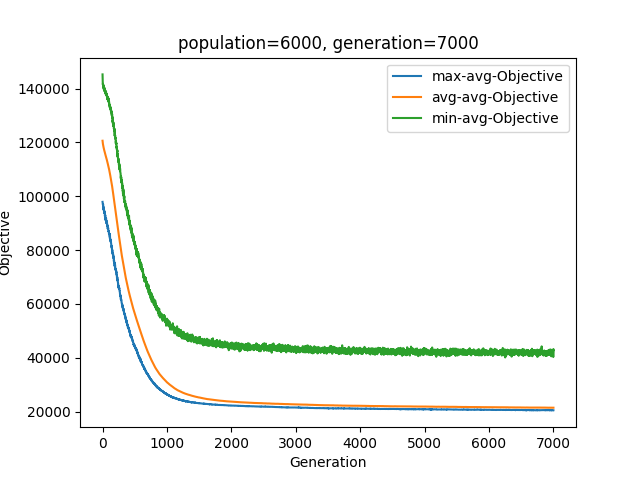


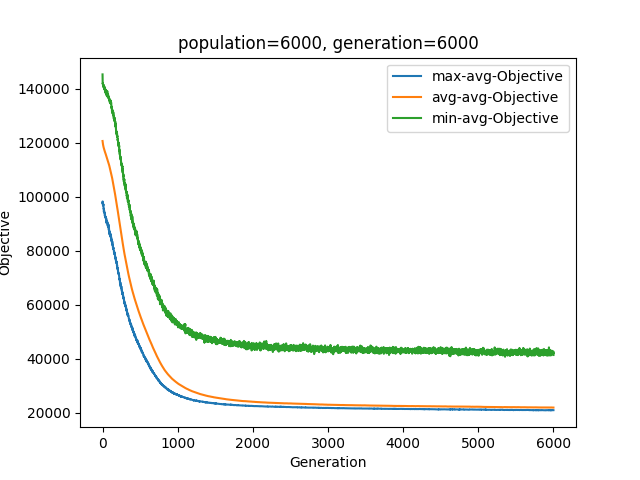
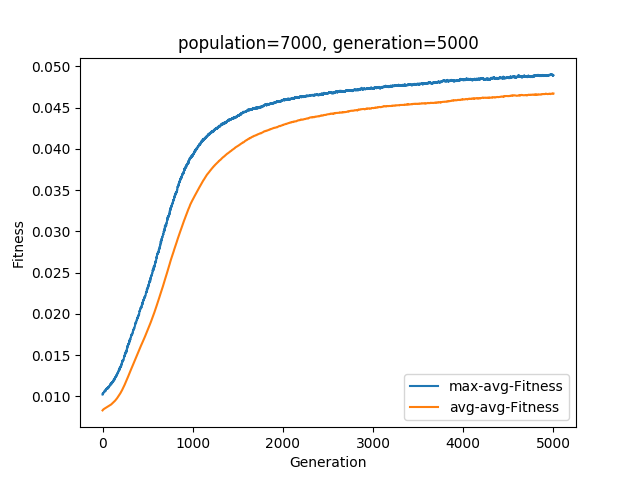
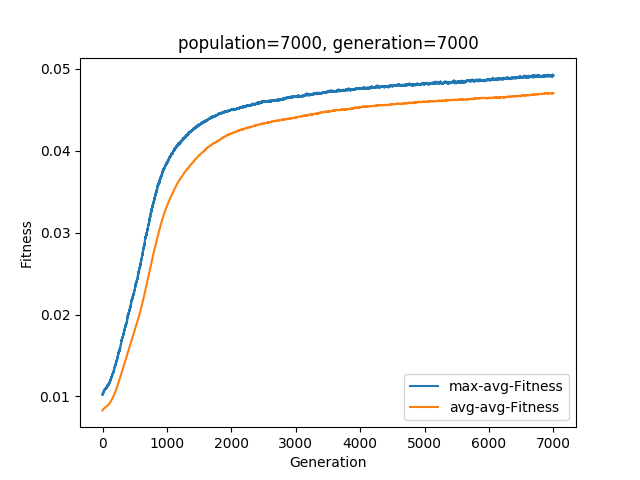


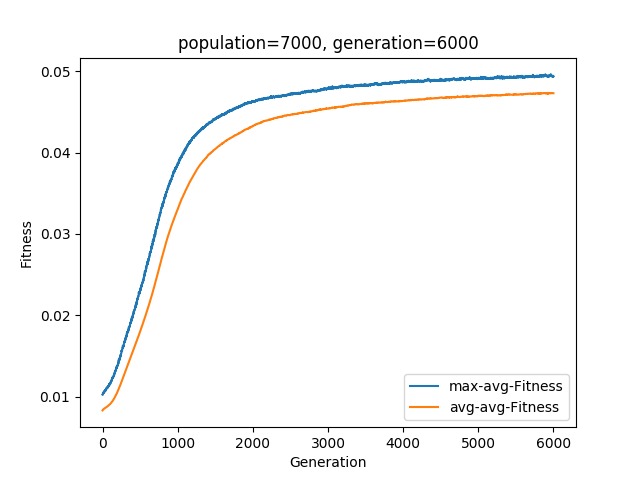
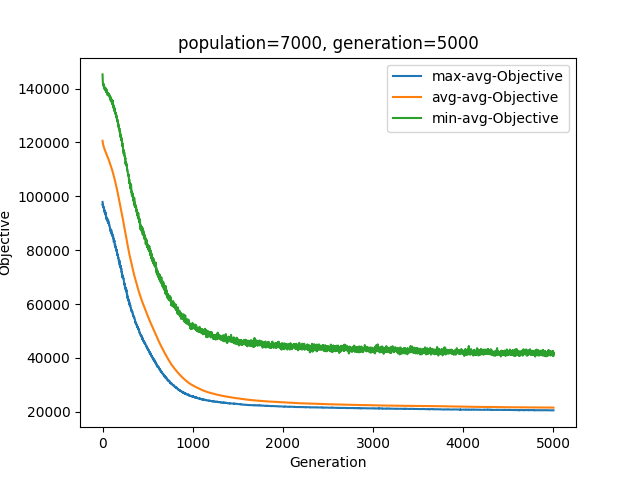
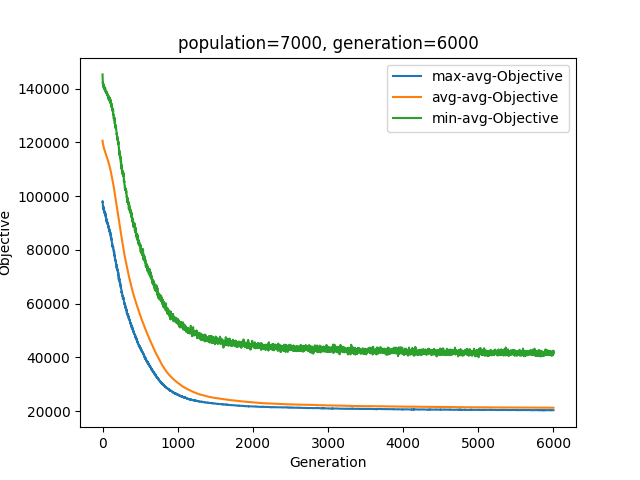
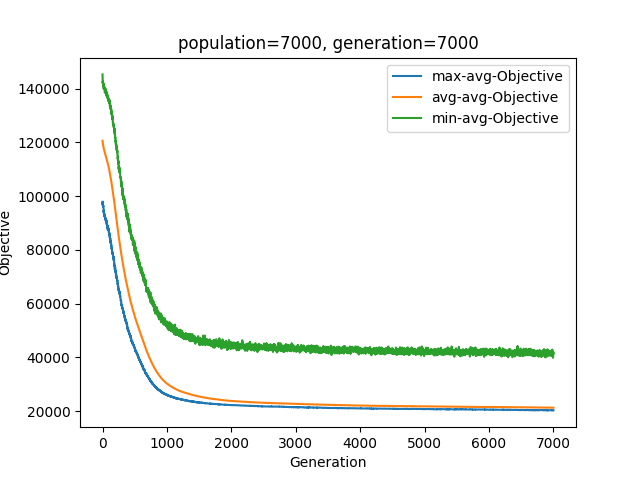


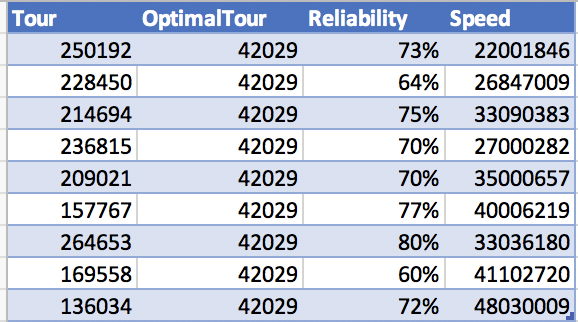






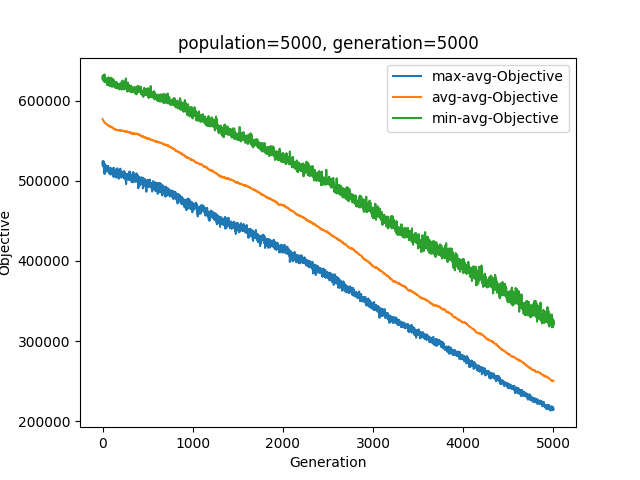
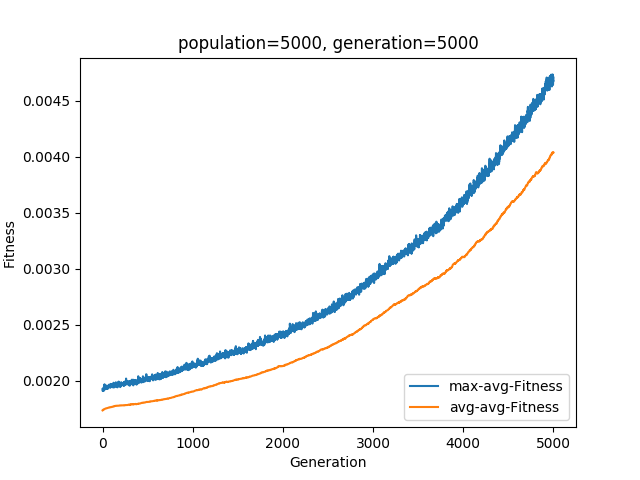


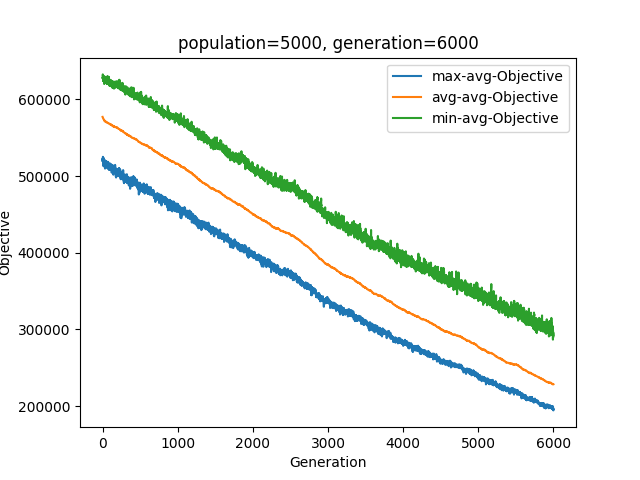
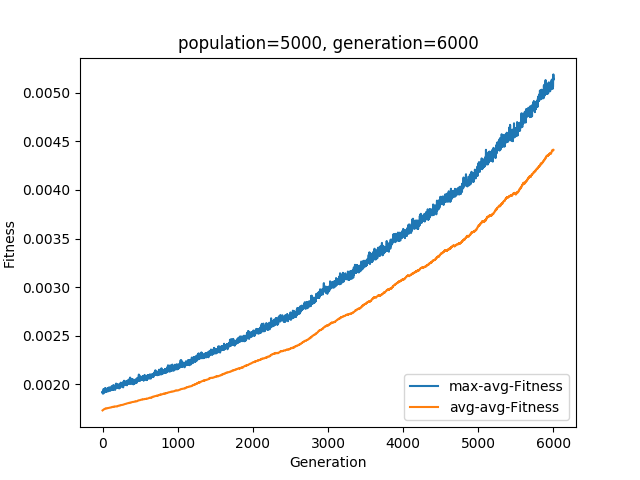


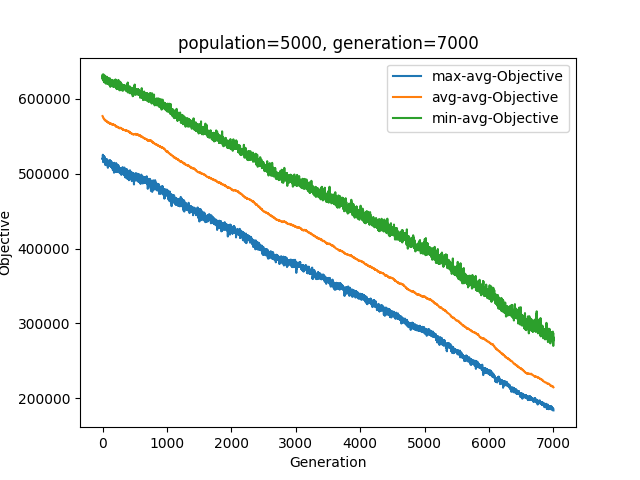


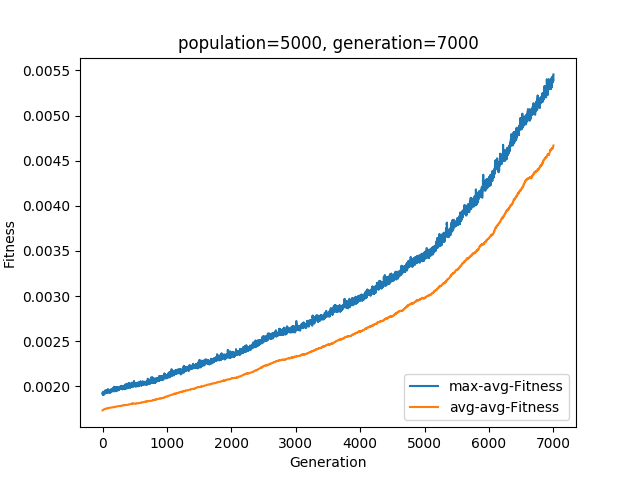
**Lin318**

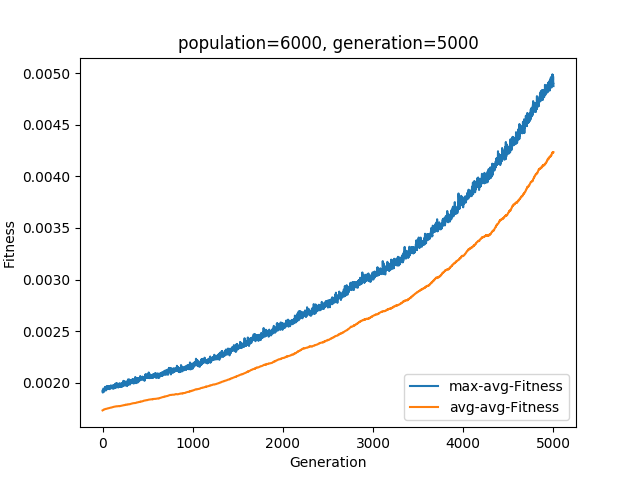
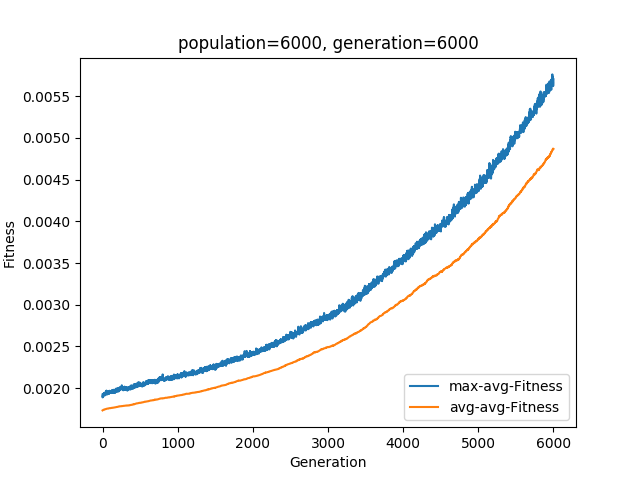
For this problem I did not put quality because it was really low. Instead I showed Tour that I obtained and optimal tour from website. As you can see my tour is considerably huge compared to known optimal value. The reason is I needed some more time as you can see from graphs they are not converging. If I hadesome more time maybe my function would converge and get optimal value.

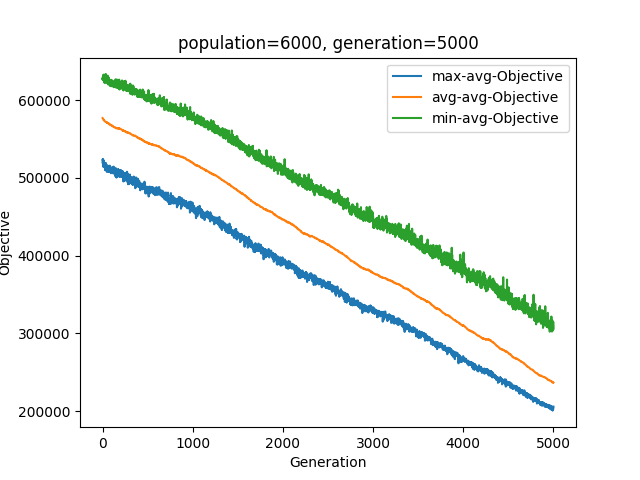
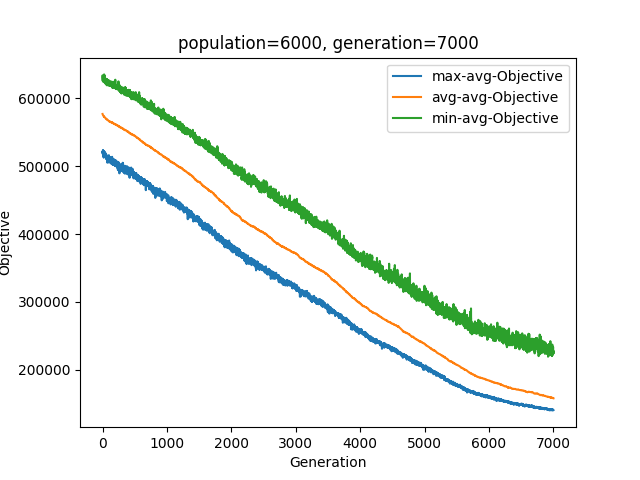


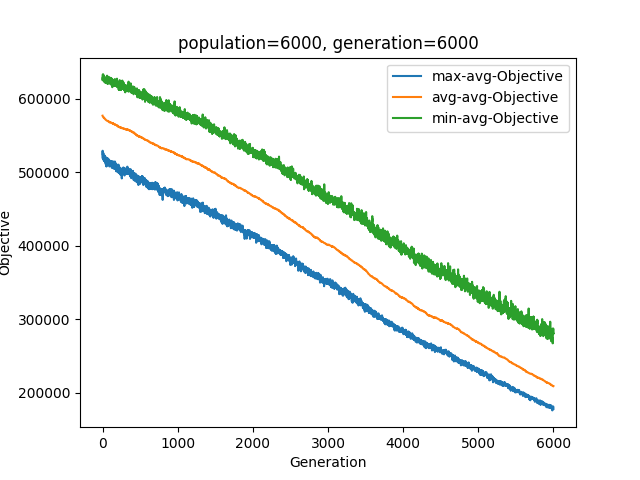
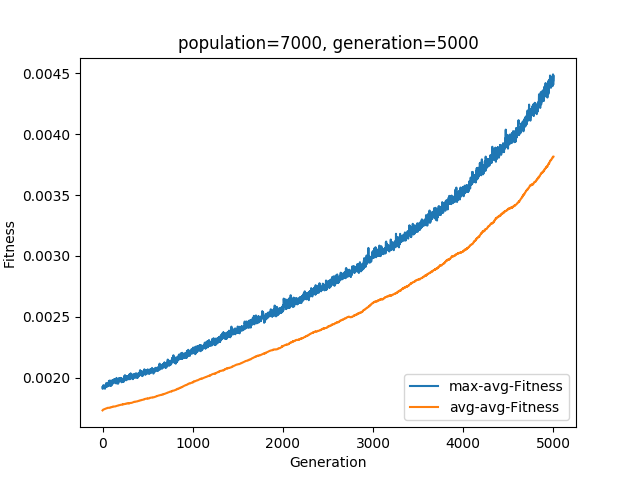
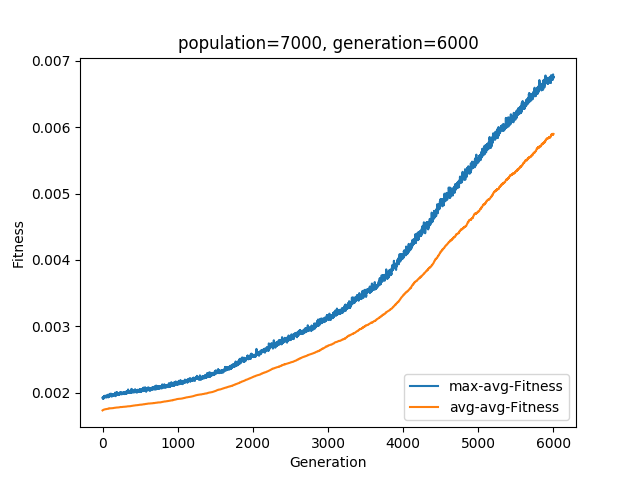
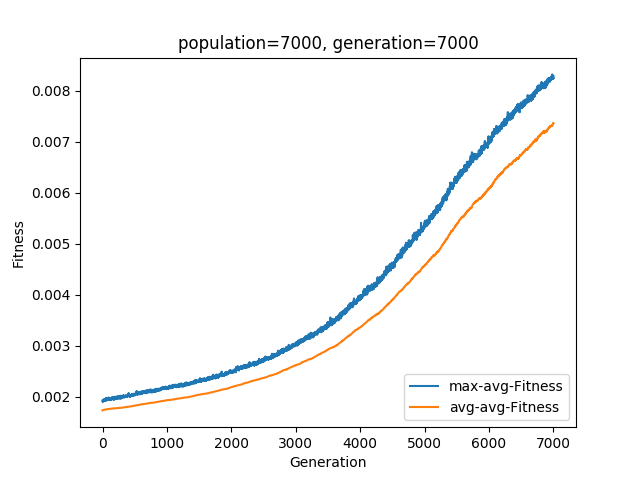


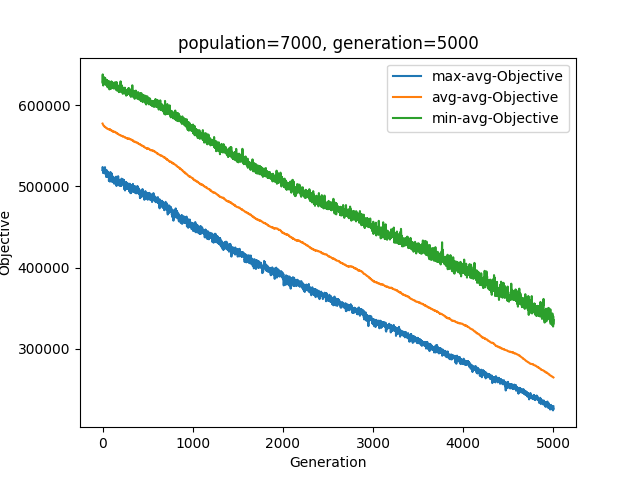
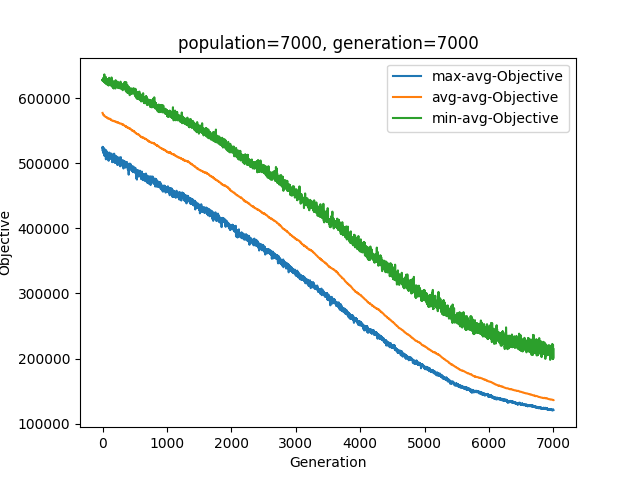
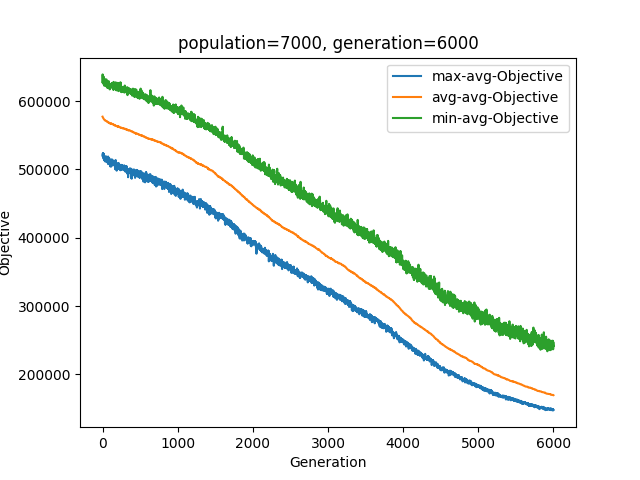












References:

[1] “Genetic Algorithms Repository” available at <https://www.cse.unr.edu/~sushil/class/gas/code/index.html>

[2] Gokturk Ucoluk “Genetic Algorithm Solution of the TSP Avoiding Special Crossover and Mutation”

[3] John McCullock “Genetic Algorithms, Mutation, Inversion mutation” available at <http://mnemstudio.org/genetic-algorithms-mutation.htm>

[4] Lee Jacobson “Applying a genetic algorithm to the traveling salesman problem” available at <http://www.theprojectspot.com/tutorial-post/applying-a-genetic-algorithm-to-the-travelling-salesman-problem/5>