Evolutionary Model
based on Fisher's Geometric Model

Daniel Duda

March 31, 2023

1 Introduction

In this report, I will describe my implementation and some properties of a partially customized evolutionary model using Python and the DEAP library. The model is based on the principles of genetic algorithms and simulates the evolution of a population over multiple generations.

2 Methodology

I have chosen DEAP library because it is a very flexible and relatively easy to use library for evolutionary modelling. Previously I tried to use the pyGAD library, but eventually it turned out that it is not possible to simulate the evolution of a population with a varying number of individuals in each generation without instancing a new population object for each generation. This is a very inefficient solution, so I decided to use DEAP instead.

1. Individual:

  - Each individual is represented by a numpy array of shape `(n_traits,)` where `n_traits` is the number of genetic traits and each trait is a float number in the range $[-1, 1]$
  - The number of traits can be changed easily in the code using the `n_traits` parameter
  - Using a numpy array instead of a list allows for vectorized operations and thus is much more efficient computationally
  - There is no sex trait. The model assumes that all individuals are hermaphrodites, but only one individual can reproduce in each generation

2. Population:

  - Initially population is represented by a list of individuals of length `init_size` and each individual has a genetic traits randomly sampled from a uniform distribution in the range $[-1, 1]$ for each trait
  - The maximum size of population is defined by the `max_size` parameter and when the population size reaches the maximum size, the individuals with the lowest fitness values are removed from the population until the population size is equal to `max_size`

3. Environmennt:

  - Environment is not implemented explicitly, but is realised by the existence of the optimal genotype (genetic traits) represented by numpy array of shape `(n_traits,)` filled with zeros at the beginning of the simulation

- During the simulation the optimal genotype is changed from generation to generation according to the formula: $\alpha(t) = \alpha(t-1) + c$ where $\alpha(t)$ is the optimal genotype at generation t and c is a random numbers sampled from a uniform distribution in the range $[0, \text{warm }\_\text{rate}]$ where `warm_rate` is a parameter of the simulation. The optimal genotype is changed in this way to simulate global warming.
- At each generation there is a 0.5% chance that the optimal genotype will be random sampled from a uniform distribution in the range $[-1, 1]$ for each trait. This is done to simulate the meteor impact.

4. Fitness:

- Fitness of each individual is calculated according to the formula:

$$\phi_\alpha(o) = \exp\left(-\frac{\|o-\alpha\|}{2\sigma^2}\right)$$

where α is the optimal genotype, o is the genotype of the individual and σ is a parameter that controls the natural selection mechanism. For $\sigma \to \infty$ the selection mechanism fades away.
- The fitness function is implemented as a vectorized function and it is applied to the entire population at once. This is much more efficient computationally.
- The value of the fitness is between the range $[0, 1]$ and provides probability of survival of the individual in the next generation explicitly.

5. Mutation:

- Mutation is implemented using a custom function that mutates individuals' traits with a probability `p_mut`
- During the mutation process, for each individual, a random number is generated from a uniform distribution. If this random number is less than `p_mut`, one of the individual's traits is mutated
- The index of the trait to be mutated is chosen randomly, and the mutation value is sampled from a normal distribution with mean 0 and standard deviation `mut_std`
- The mutation value is then added to the selected trait, resulting in a new trait value for the individual
- This process enables the exploration of the search space and introduces genetic diversity into the population

6. Parent selection:

- Parent selection is performed using a custom function that filters individuals based on their fitness values and a competition rate `comp_rate`
- For each individual, a random number is generated from a uniform distribution. If this random number is less than the product of $(1 -$ `comp_rate`$)$ and the individual's fitness value, the individual is selected as a parent
- This selection method favors individuals with higher fitness values, but it also allows individuals with lower fitness values to have a chance of being selected as parents, promoting diversity in the population
- The selected parents are then used to create offspring in the reproduction process

7. Reproduction:

- The parents are randomly paired up to create offspring; each pair of parents produces one child
- For each pair of parents, the child is created by taking the element-wise average of the parent's genetic traits with a random scaling factor between 2 and 4

3 Results

- **Optimal mutation probability and effect for adaptation:**
  For the given `_CONFIG` parameters:
  - `init_size`: 100
  - `max_size`: 1000
  - `n_generations`: 50
  - `n_traits`: 3
  - `trait_min`: -1
  - `trait_max`: 1
  - `init_opt_gen`: `np.zeros(3)`
  - `sel_std`: 0.95
  - `warm_rate`: 0.01
  - `competition_rate`: 0.1
  - `seed`: 2137

  I run the simulation 5 times for `p_mut` and `mut_std`$\in$ `numpy.linspace(0.1, 0.9, 10)` and calculated the average survival rate for each combination of `p_mut` and `mut_std` by summing the generations where there was at least one individual and dividing it by the number of simulation runs.

  This is the result:

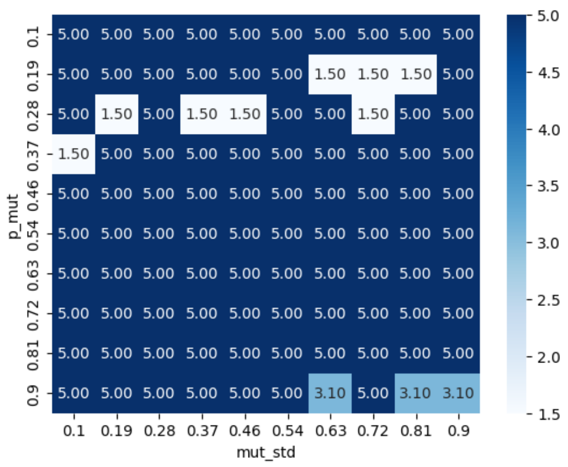

Figure 1: Optimal `p_mut` and `mut_std`

- **Visualization of evolution in time:**
  For the given `_CONFIG` parameters as befor but with `p_mut` $= 0.5$ and `mut_std` $= 0.4$ I run the simulation for 26 generations.

This is the result:

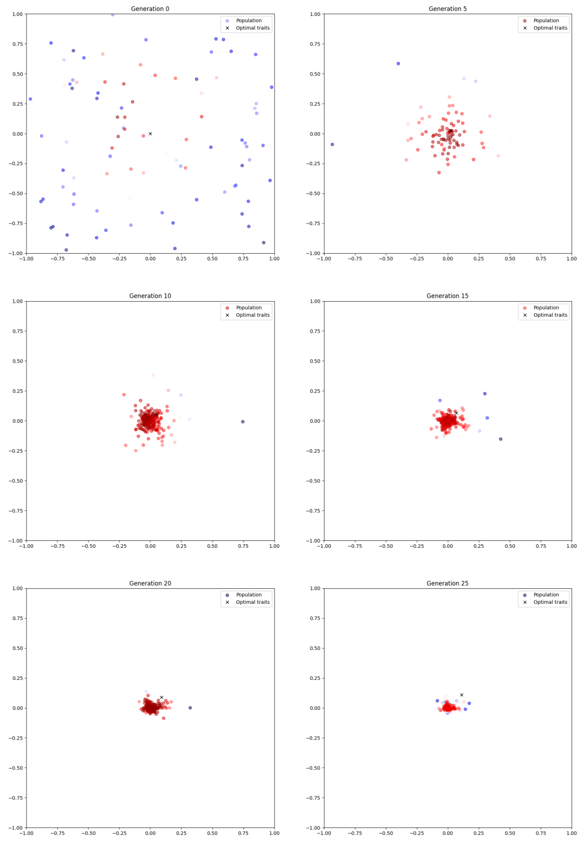

Figure 2: Evolution of population using PCA for 2 components

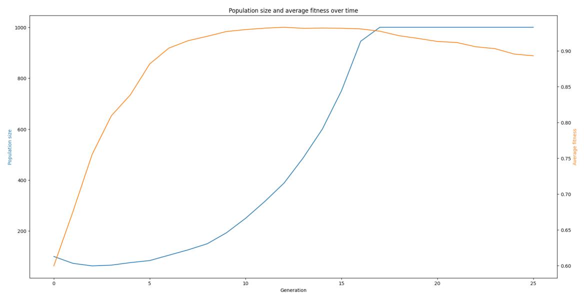

Figure 3: Population size and average fitness over time

4 Conclusion

As seen in results section, my model is very unstable for a bigger number of generations. It may be due to the fact that the after a certain number of generations the optimal traits can go out of the range of the traits of population. Also events like meteor impact can cause the extinction of the population sooner or later. I think that the model can be improved by adding some kind of a mechanism that prevents the optimal traits from going out of the range of the traits of the population.