Politecnico di Milano
Gaia Biasolo, Matteo Giuseppe Cigada, Federico D'Agostini

Lab 06: Genetic algorithms

24/10/2025

Abstract

This report details the implementation and sensitivity analysis of a genetic algorithm (GA) to minimize a two-variable function. The study investigated the influence of bit resolution, maximum iterations, and population size on convergence. Optimal results were achieved only through the combined use of mutation and elitism.

Contents

1. Introduction

The aim of the assignment is to develop and use a genetic algorithm to solve a simple problem, where a single objective function is considered.

The genetic algorithm is developed step by step, highlighting the function and purpose of each element that constitutes its structure.

1.1 Description of the problem

The requirement for this lab is to implement a genetic algorithm to minimize the function:

$$y = {x_1}^2 - {x_2}^2 \quad \text{with } x_1, x_2 = [0\ 5]$$

In order to do so, the minimization problem is reformulated as a maximization one subtracting the value of the function to a much larger one (in our case, from an analytical analysis of the problem, 100 was given). So, the objective function becomes:

$$y = 100 - ({x_1}^2 - {x_2}^2)$$

As far as the data for the algorithm is concerned, the two variables have to be coded in 16 bits and the population consists of 100 members.

2. Description of the algorithm

2.1 Baseline case

The request of the lab was to implement the algorithm step by step. The baseline case from which the development started is described by the following steps(**Error! Reference source not found.**)

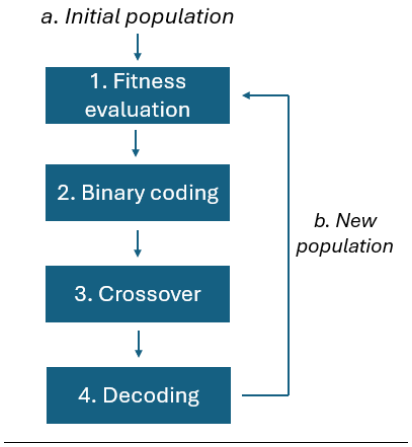

Figure 1

a. *Initial population:* the members of this population are randomly selected in the variable space defined by the problem.
1. **Fitness evaluation**: fitness is a parameter that is strictly related to the objective function of the problem, and it is used to determine how good an individual is. As far as this lab is concerned, the fitness coincides with the objective function.
2. **Binary coding**: this passage is required by the algorithm to get a string of binary values that can be easily manipulated to create changes and a new population. The two variables are coded and united in a single string.
3. **Crossover**: this phase is the crucial step of a genetic algorithm. One bit is randomly picked and the strings are split and recombined to get a new population (children) that is derived from the previous one (parents). However, this process is stochastic but not entirely random: specifically, for each member of the population, fitness is evaluated, and a percentage is associated to each parent by dividing it by the sum of all the fitness values of the population. A Roulette wheel selection is then implemented, assigning to each member of the population a region on the wheel proportional to the percentage previously evaluated. The members that will undergo crossover are then selected "spinning this wheel"; this ensures that the parents that are most likely to have an offspring are also the ones with a higher fitness value. Depending on the implementation of the algorithm, it can happen that not all the

parents that are selected undergo crossover and pass on to the next generation without changes. In this case nonetheless, the crossover probability was set as 100%.

4. **Decoding**: the new population must be decoded for the algorithm to start again evaluating the fitness.

2.2 Mutation

In the second point, the only change applied was the implementation of the effect of a mutation as shown in Figure 2**Error! Reference source not found.**.

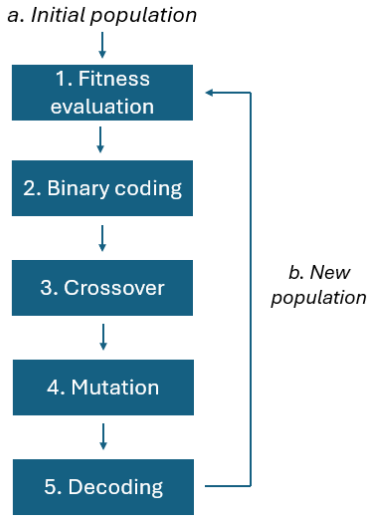

Figure 2

A mutation is the change of a random bit that is used to avoid the algorithm reaching the local minimum. This could be useful for a small population. Practically, it adds random noise to the algorithm.

A mutation parameter is defined. This parameter identifies the probability of the members of the population to undergo mutation. It is defined as

$$p_{mut} = par * p_{min} + (1 - par) * p_{max}$$

Where p_{min} is the minimum between the inverse of the number of members of the population and the inverse of the length of each member (the same for p_{max}).

In addition, in this case, $par = 0.5$.

The aim of adding the mutation is to avoid the algorithm reaching a local minimum, introducing at each iteration some random changes that could allow the algorithm to find a better solution genetically far from the current ones. A disadvantage of this additional step

is that the algorithm reaches convergence more slowly. If the probability of mutation is too high then the mutation does not help the algorithm at all but may prevent convergence and reaching an optimal solution.

2.3 Discard

In the third point, the request was to implement a discard function as showed in Figure 3.

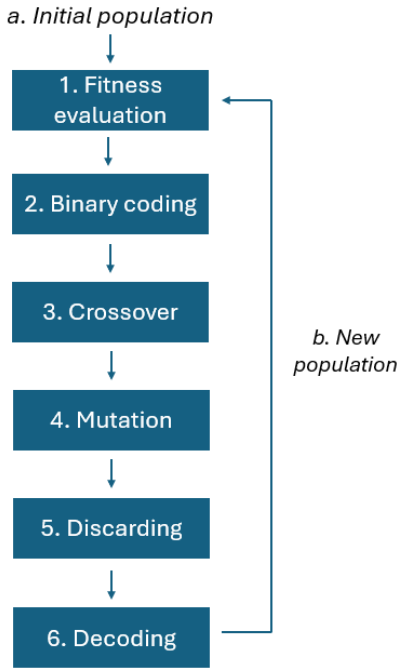

Figure 3

The discard function was added partly to exploit all the advantages given also by the mutation step avoiding the disadvantages introduced by it. Since the modifications applied by both the crossover and the mutation steps are random, the parents may be fitter compared to the offspring. The main objective of the discard function is to evaluate the best fitting members of the population between parents, children and modified children. By keeping only the members with the highest fitness, convergence is ensured in less iterations.

The function was implemented by grouping parents, children and modified children, evaluating their fitness and sorting them from the best to the worst fitting. Only the first 100 were chosen as part of the next population.

3. Sensitivity analysis on algorithm parameters

The effect of varying the number of bits, the number of max iterations and the population size on the computed result have been investigated. The reference case is:

- nbit = 16
- population size = 100
- max iterations = 200

One parameter at a time has been varied while keeping the other ones fixed; the following results have been obtained.

3.1 Effect of number of bits

The number bits tested were 8,16,32.

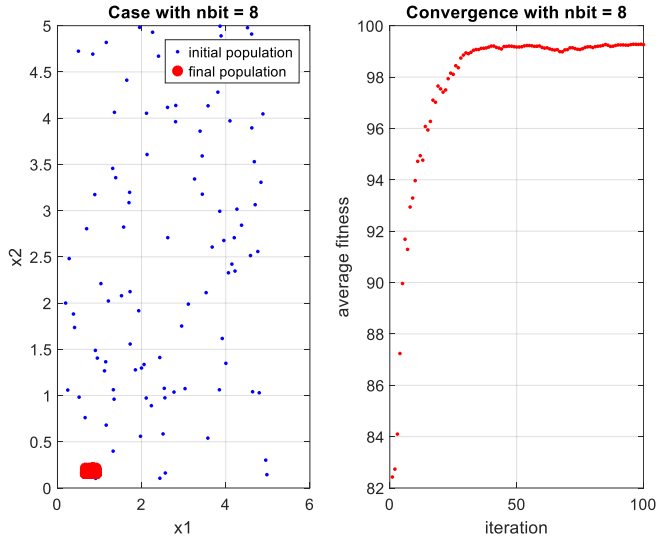

Figure 4

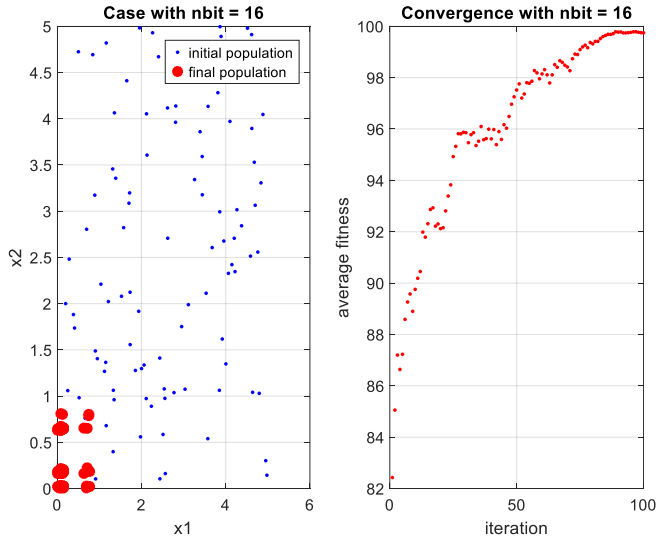

Figure 5

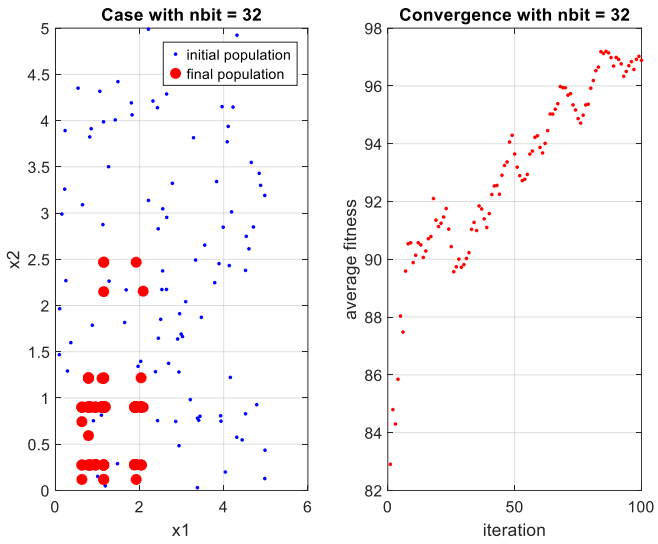

Figure 6

With an increasing number of bits, a larger number of genetic codes can be created inside the feasible domain (a larger number of "chromosomes"). By looking at the results, it may appear that, as the number of bits increases, the final population is more dispersed in the x1, x2 plane. However, it is important to remember that this is the result of a single run and each new run will provide a different result; for this reason, this observation cannot be taken as a general conclusion that is always going to be verified.

3.2 Effect of number of max iterations

The max number of iterations tested were 50,100,200.

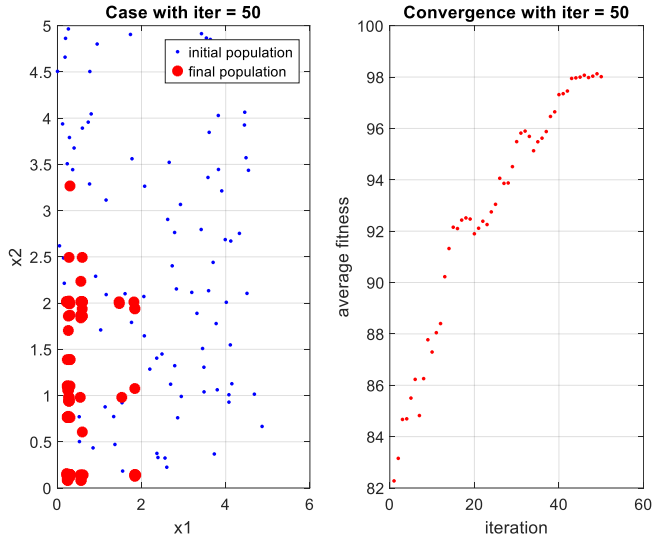

Figure 7

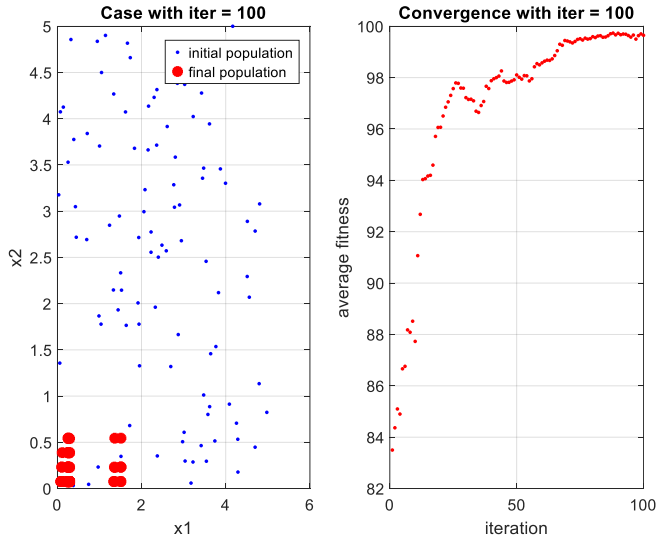

Figure 8

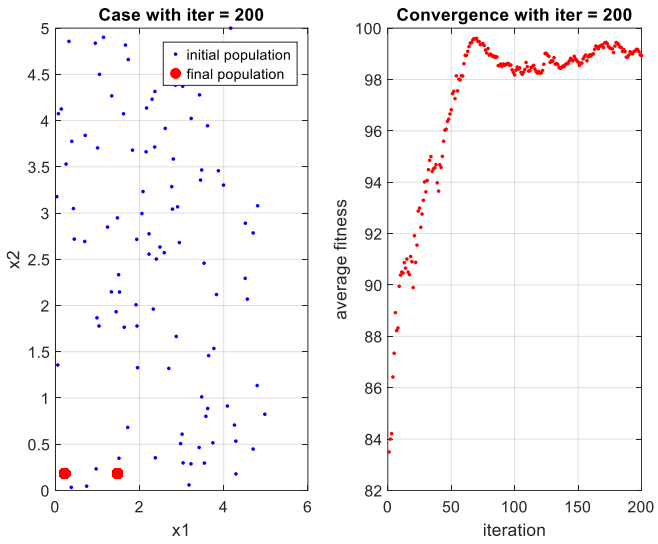

Figure 9

The number of max iterations performed has a more significant result on the simulation output. As a matter of fact, if the number of generations increases, the same genetic trait with higher fitness will spread across more individuals. For instance, when only 50 generations are considered, the final population is widespread in the x1, x2 plane. As the number of generations considered increases, the individuals of the final population tend to be more similar to each other (the points get closer in the x1, x2 plane). Since at this point of discussion no elitism has been introduced yet, the effect of increasing the number of generations on the fitness is not monotonic, in some generations the children may have a lower fitness than the parents.

3.3 Effect of population size

The population size tested are 50,100,200.

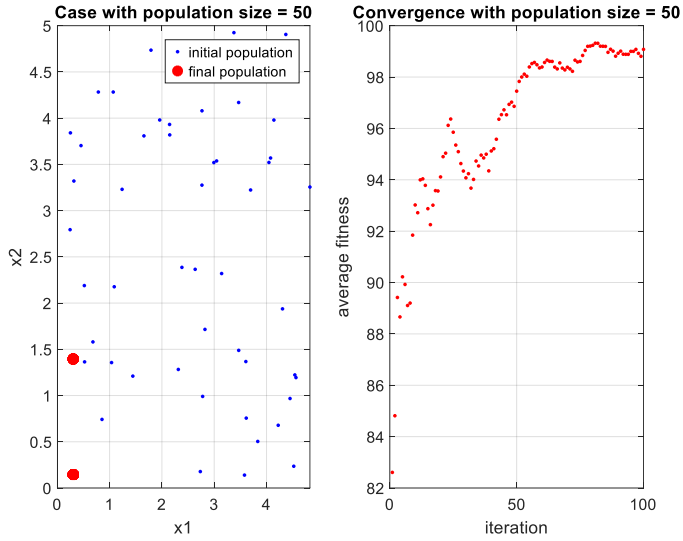

Figure 10

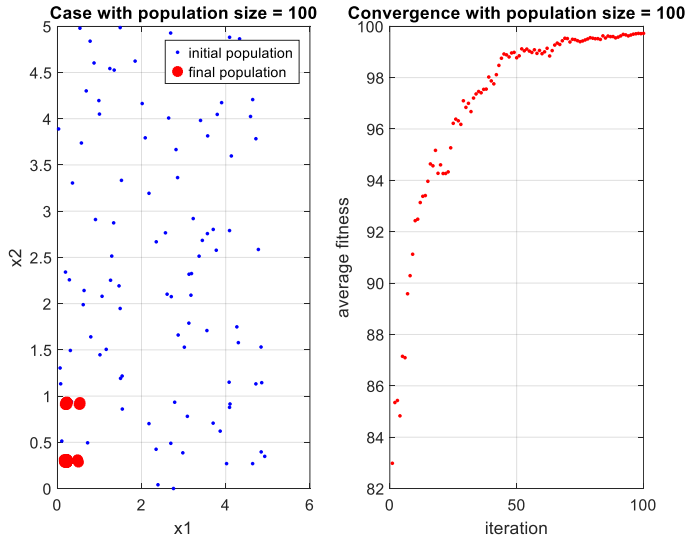

Figure 11

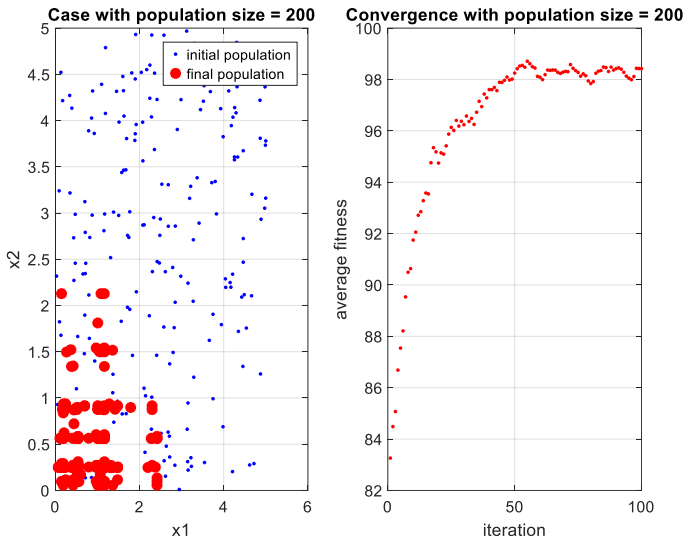

Figure 12

As expected, if the population size increases without changing the number of generations considered, the individuals of the final population will be more spread out in the x1,x2 plane. Having a larger population increases the chances of finding a global minimum, however, more iterations are required for the fitness to increase; this is why the simulation performed with 200 individuals has the lowest average fitness.

3.4 Effect of mutation

In the results presented below, the reference settings described previously have been used, therefore, the only reasonable comparison that can be made is with the baseline case Figure 5.

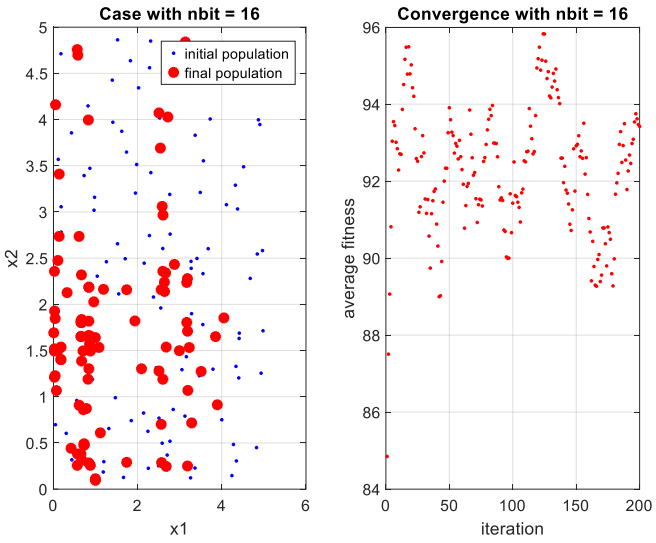

Figure 13

Introducing mutation without elitism makes the result much worse; for instance, recombining genes randomly without any further modifications to the algorithm severely penalizes the convergence. As expected, the final population is widespread in the x1, x2 plane and the average fitness has a strong non monotonic trend (differently from the baseline case).

3.5 Effect of discarding/elitism

In the results presented below the reference settings described previously have been used; therefore, the only reasonable comparison that can be made is with the baseline case Figure 5. Note that usually a simulation is ended as soon as the whole population has reached the highest level of fitness, in this case, to perform a better comparison with the previous examples, the number of iterations has been imposed.

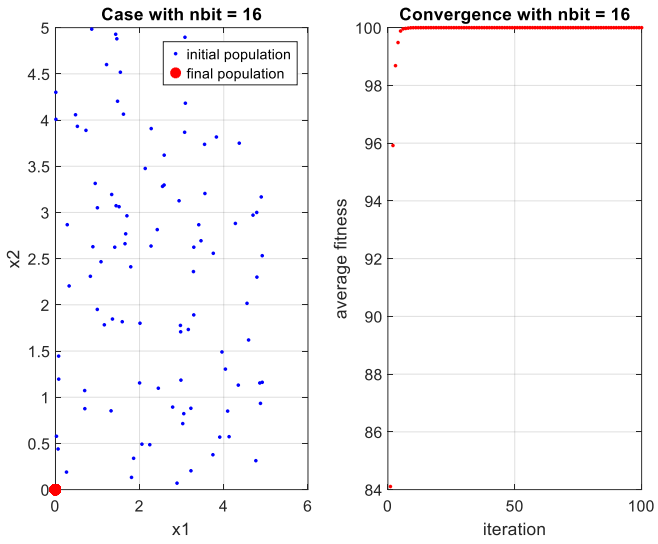

Figure 14

The combination of mutation + discarding/elitism provides excellent results, all the individuals of the final population coincide with the minimum of the function, and the average fitness of the populations reaches the maximum value in a very limited number of iterations.

4. Conclusion

This laboratory implemented a genetic algorithm (GA) to solve a minimization problem. The analysis focused on the step-by-step development of the algorithm and a sensitivity analysis of key parameters such as the number of bits, the maximum iterations, the population size. Subsequently, the effects of mutation and elitism were analysed.

1. **Effect of number of bits:** increasing the number of bits (from 8 to 32) increases the "resolution" or the number of possible "chromosomes." While this theoretically improves the chances of finding the global optimum, it also leads to a more dispersed final population in a single run. However, this last observation cannot be taken as an absolute conclusion that will always be verified because each run provides a different result.
2. **Effect of number of max iterations:** with a low number of generations considered (50), the population remains widespread. As the max iterations increase (up to 200), the population converges more tightly, reflecting the successful propagation of high-fitness genetic traits. However, without elitism, the fitness trend remains non-monotonic, as parents can still produce lower-fitness children.
3. **Effect of population size:** larger populations require significantly more iterations to achieve a comparable average fitness level.
4. **Effect of discarding/elitism:** The introduction of mutation alone severely degrades the algorithm's performance, causing the final population to be highly dispersed and the average fitness trend to become strongly non-monotonic. This confirms that introducing random noise without a mechanism to preserve quality, compromises the converge. In contrast, the combination of mutation and discarding/elitism yields the best results. Elitism acts as a powerful selection tool, ensuring that the fittest individuals always survived to the next generation. This configuration results in rapid convergence.