KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

Deemed to be University U/S 3 of the UGC Act, 1956

Genetic Algorithm Report

Name: Sunali Patro

Roll: 21052628

Section: CS 18

Acknowledgement

With immense please I, Miss Sunali Patro presenting “ Genetic Algorithm report” as part of the curriculum. I wish to thanks all the people who gave me unending support.

I express my profound thanks to Sohail Khan . And all those who have indirectly guided and helped me in preparation of this report.

Sunali Patro

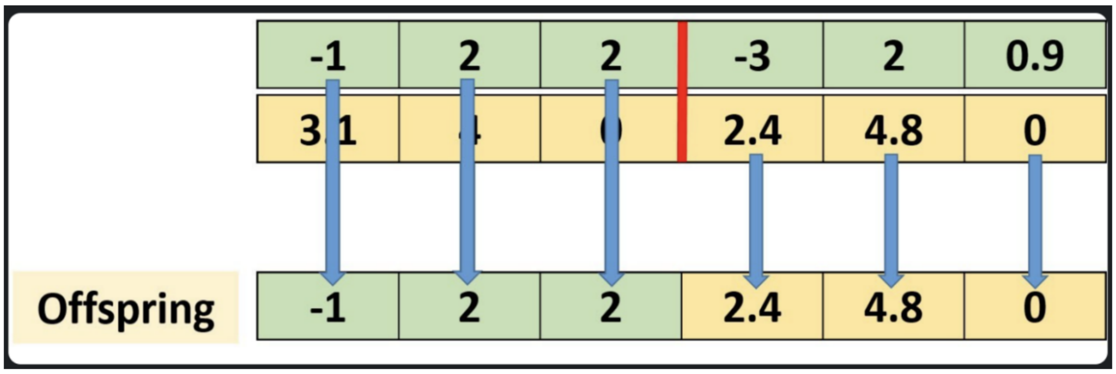
Abstract

The genetic algorithm (GA) starts from a randomly generated image of the same shape as the input image. This randomly generated image is evolved, using crossover and mutation, using GA until it reproduces an image similar to the original one. The exact original image might not be accurately reproduced, but at least a similar image will be generated.

Crossover

Mating 2 organisms means creating a new offspring that shares the genes inside both of them. The crossover operation selects a number of genes from each parent and places them into their offspring.

The next figure shows an example in which single-point crossover is applied between 2 parents in order to create an offspring. Single-point crossover works by selecting a point at the chromosome. Genes before this point are selected from one parent, and genes after it is selected from the other parent. As a result, the offspring shares genes from both parents.



The crossover operation is applied in the project using a function named crossover(). It accepts 3 arguments: the parents selected previously using
the select_mating_pool() function, the input image shape (img_shape), and the number of offspring to return (n_individuals), which defaults to 8.
Let’s assume that all solutions in a given population share a gene that represents a bad property. After applying crossover, this gene will definitely be available in the offspring. If the offspring takes its genes from 2 parents where each parent has a bad gene, then the offspring will now have 2 bad genes.
As a result, it will be worse than its parents. This is why it’s preferable that the next generation keep the previous parents in addition to the generated offspring. Even if the offspring are worse than the parents, the parents will be kept to avoid moving GA toward solutions that aren’t evolved. This is

why the crossover() function returns the new population, which consists of both the current parents and their offspring.
But what do we do if the offspring is always worse than the parent? There is an operation called mutation that's applied after crossover in order to introduce modifications to the offspring.
These modifications might solve a problem in the parents by replacing a bad gene with a better one. It might also introduce a problem by replacing a good gene and making it worse. If that happens, then the parents kept previously will prevent the selection of these bad offspring.

Mutation

The mutation operation selects some genes within the chromosome and then randomly changes their values.

It’s implemented according to the mutation() function listed below. It accepts the population returned by the crossover() function, number of parents, and the percent of the genes to be changed. The number of parents is passed in order to simply apply the mutation over the offspring and skip the parents.

Avoid setting the percentage to a high value because it will introduce many random changes, which will definitely lead to bad solutions. Slight changes using small percentages might (but not certainly) introduce good changes.

The previous figure applied the crossover and returned an offspring to which the mutation is applied, as shown in the next figure. Assuming

that a single gene is selected for mutation, its value is changed by randomly generating a number that is added to it.
The result of this addition will replace the previous gene value. The random change may be in other forms than in addition. In this project, as implemented in the mutation() function, the random change is made by replacing the previous values with a new value within the range 0–255. This range is selected because the images are represented as 8-bit unsigned integers.

Selection

The parents selected from a given population are the best solutions within it. When we say “best solutions”, we’re referring to the solutions with the highest fitness values.
Assume that the population has 6 solutions and their fitness values are as given in the figure below. Before selecting the best parents, we need to decide how many parents to select. Assuming that half (3 out of 6) of the solutions will be selected, then the best 3 solutions are the ones with the highest fitness values. This why the solutions with ID 4, 2, and 6 are selected.
The parents are returned in this project according to a function named select_mating_pool() (shown below). It accepts 3 arguments: population (pop), the fitness values (qualities), and the number of parents (num_parents). It loops through the parents to select the ones with the highest

fitness values and return them into an array named parents.
The function works by searching for the solution with the maximum fitness value and returning it into the parents array. In order to avoid selecting this parent again in the next iteration of the loop, its fitness value is set to -1. This guarantees not selecting it again.

Thank You.