Artificial Intelligence (AI)

Lab Sheet No: 6

Introduction

A genetic algorithm is a search heuristic that is inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.

Notion of Natural Selection

The process of natural selection starts with the selection of fittest individuals from a population. They produce offspring which inherit the characteristics of the parents and will be added to the next generation. If parents have better fitness, their offspring will be better than parents and have a better chance at surviving. This process keeps on iterating and at the end, a generation with the fittest individuals will be found.
This notion can be applied for a search problem. We consider a set of solutions for a problem and select the set of best ones out of them.
Five phases are considered in a genetic algorithm.

1. Initial population
2. Fitness function
3. Selection
4. Crossover
5. Mutation

Initial Population

The process begins with a set of individuals which is called a Population. Each individual is a solution to the problem you want to solve. An individual is characterized by a set of parameters (variables) known as Genes. Genes are joined into a string to form a Chromosome (solution).

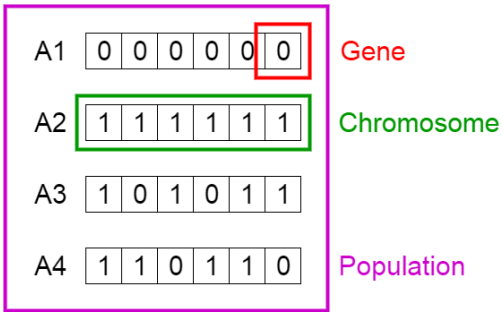

Figure 1: Population, Chromosomes and Genes

In a genetic algorithm, the set of genes of an individual is represented using a string, in terms of an alphabet. Usually, binary values are used (string of 1s and 0s). We say that we encode the genes in a chromosome.

Fitness Function

The fitness function determines how fit an individual is (the ability of an individual to compete with other individuals). It gives a fitness score to each individual. The probability that an individual will be selected for reproduction is based on its fitness score.

Selection

The idea of selection phase is to select the fittest individuals and let them pass their genes to the next generation.

Two pairs of individuals (parents) are selected based on their fitness scores. Individuals with high fitness have more chance to be selected for reproduction.

Crossover

Crossover is the most significant phase in a genetic algorithm. For each pair of parents to be mated, a crossover point is chosen at random from within the genes.

For example, consider the crossover point to be 3 as shown below.

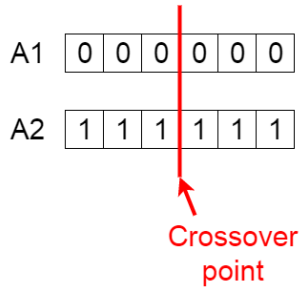

Figure 2: Crossover point

Offspring are created by exchanging the genes of parents among themselves until the crossover point is reached.

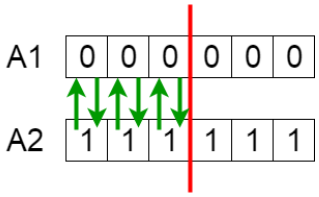

Figure 3: Exchanging genes among parents

The new offspring are added to the population.

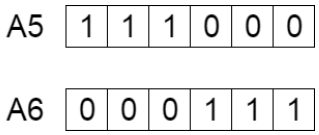

Figure 4: New offspring

Mutation

In certain new offspring formed, some of their genes can be subjected to a mutation with a low random probability. This implies that some of the bits in the bit string can be flipped.

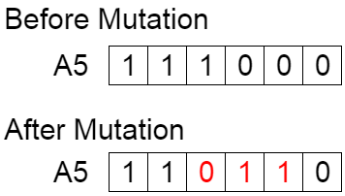

Figure 5: Mutation: Before and After

Mutation occurs to maintain diversity within the population and prevent premature convergence.

Termination

The algorithm terminates if the population has converged (does not produce offspring which are significantly different from the previous generation). Then it is said that the genetic algorithm has provided a set of solutions to our problem.

Scenario and Problem to be solved

Let's imagine, we have a painter robot similar to the robot which picked up cans in the lectures. We will use this robot to paint the floor of a room. To make it interesting, the painter starts at a random place in the room, and paints continuously. We will also imagine that there is exactly enough paint to cover the floor. This means that it is wasteful to visit the same spot more than once or to stay in the same place. To see if there is a optimal set of rules for the painter to follow, you will create a genetic algorithm. You may write your own code from scratch or use painter_play.m or painter_play.py as starting points.

As inputs, this function receives

1. A chromosome: A 1x54 array of numbers between 0 and 3 that shows how to respond (0: no turn, 1:turn left, 2:turn right, 3: random turn left/right) in each of the 54 possible states. The state is the state of the squares forward/left/right and the current square. Let [c, f, l, r] denote states of the current square, forward square, left square and right square respectively. Write 0 for empty, 1 for wall/obstruction and 2 for painted.
   Note that $c \in \{0, 2\}$ and $f, l, r \in \{0, 1, 2\}$ so there are $2 \times 3^3 = 54$ possible states.
2. An environment: A 2D array representing a rectangular room. Empty (paintable) space is represented by a zero, while furniture or extra walls are represented by ones. Outside walls are automatically created by painter_play().

The function painter_play() then uses the rule set to guide a painter, initially placed in the room with a random position and direction, until the paint can is empty. Note that the painter does not move when it tries to walk into a wall or furniture. The efficiency (total fraction of paintable space covered) is then given as an output, as well as the X-Y trajectory (i.e. the positions of the painter at each time step) of the painter. To see that the painter works, you can try passing it an empty room for an environment and a trivial chromosome. For example, a chromosome consisting of all 3s produces a kind of random walk. Now do the following:

Assignment (1): Think of a simple strategy for the painter to cover a lot of space in an empty room. Describe this strategy in a few words or sketch it, but do not try to encode it in the chromosome.

Assignment (2): Create 50 random chromosomes in a 50x54 matrix, as well as a 20x40 empty room. Create a genetic algorithm to evolve this population over 200 generations, playing each chromosome several times and storing the chromosomes average efficiency as the fitness.

You may choose any rule for picking the next generation from the previous one so long as it includes crossovers and mutation and that individuals with higher fitness are more likely to have offspring in next generation. (An example is to use single-point crossover with a mutation rate of 0.002 per locus per generation.) Plot the final set of chromosomes. Plot an example trajectory of one of the more successful chromosomes (or make a video). Is this what you expected?

Assignment (3): Plot the average fitness in the population vs generation. You will likely see large sudden jumps in fitness, corresponding to strategic innovations. In your own words, write down two possible examples of an innovation that would increase fitness.

Assignment (4): Add some furniture to the empty room (about 100 square metres in total) and use one of your highly evolved chromosomes, and plot the trajectory (or make a video). How does the efficiency compare to that in an empty room? If the strategy fails, how does it fail? Now try running the genetic algorithm with your new furnished room from the start. How does the strategy compare to the empty room strategy?